Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten

Dissertation zur Erlangung des Doktorgrades der Fakultät für Angewandte Wissenschaften der Albert-Ludwigs-Universität Freiburg im Breisgau

Klaus Dorer
Institut für Informatik und Gesellschaft
Abteilung Kognitionswissenschaft
Albert-Ludwigs-Universität Freiburg

20. Dezember, 1999

Dekan der Fakultät:

Prof. Dr. Gerald Urban
Institut für Mikrosystemtechnik
Albert-Ludwigs-Universität Freiburg
Am Flughafen 17
D-79110 Freiburg

Referenten:

Prof. Dr. Bernhard Nebel
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Am Flughafen 17
D-79110 Freiburg

Prof. Dr. Gerhard Strube
Institut für Informatik und Gesellschaft
Abteilung Kognitionswissenschaft
Albert-Ludwigs-Universität Freiburg
Friedrichstraße 50
D-79098 Freiburg

20. Dezember 1999

Danksagung

Diese Dissertation wurde durch ein Stipendium der Deutschen Forschungsgesellschaft im Rahmen des Graduiertenkollegs *Menschliche und Maschinelle Intelligenz* gefördert. Neben der Teilnahme an den im Rahmen des Graduiertenkollegs veranstalteten Ringvorlesungen, Frühjahrstreffen und Herbstschulen, wurde so auch die Teilnahme an mehreren nationalen und einer internationalen Konferenz ermöglicht.

Mein herzlicher Dank gilt meinem Betreuer *Prof. Dr. Gerhard Strube*, der immer Zeit fand, meine Fragen zu beantworten und Probleme zu diskutieren. Mit seinem profunden Wissen, sowohl in der Kognitionswissenschaft als auch in der Informatik, trug er wesentlich zum Gelingen der Arbeit bei. Auch scheute er keinen Aufwand, um mir eine optimale Arbeitsumgebung zur Verfügung zu stellen. Es wird mir lebhaft in Erinnerung bleiben, daß er mir seinen eigenen Computer anbot, als sich herausstellte, daß mein PC zu langsam für die RoboCup-Fußballumgebung war. Da auch dieser zu langsam war, wurde, trotz knapper Mittel, ein neuer PC angeschafft.

Ebenso danke ich meinem fachlichen Betreuer *Prof. Dr. Bernhard Nebel* für das zielsichere Aufspüren von Schwächen und Fehlern, das der Qualität der Arbeit sehr zuträglich war.

Weiterhin danke ich der ganzen Abteilung Kognitionswissenschaft sowie Tobby Tyrrell, Phil Goetz und Tim Norman für wertvolle Diskussionen und Anregungen.

Schließlich gilt mein besonderer Dank meiner Frau, die die vielen Entbehrungen einer Wochenendbeziehung mit Geduld ertrug, und es darüber hinaus immer wieder verstand, mich aufzubauen und zu motivieren. Ebenso danke ich meinen Eltern und Schwiegereltern für die unschätzbare moralische Unterstützung. Besondere Erwähnung verdient nicht zuletzt das mühevolle Korrekturlesen der Dissertation von Ute Ongyert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Der Begriff des autonomen Agenten hat in den letzten Jahren über die Grenzen der künstlichen Intelligenz Forschung hinaus zunehmend an Bedeutung gewonnen. In verschiedenen Bereichen der Informatik sind autonome Agenten Gegenstand der Forschung. Neben zahlreichen Workshops im Rahmen von Konferenzen gibt es eigene Konferenzen zum Thema autonome Agenten. So findet seit 1997 jährlich die Autonomous Agents Konferenz statt, seit 1996 die International Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology, um nur zwei Beispiele zu nennen.

Daneben finden sich Agenten bereits in zahlreichen Anwendungen wieder. Das gilt im besonderen im Bereich der virtuellen Agenten, der Soft-Bots oder Shopping Agenten. Aber auch reale Agenten (Roboter) werden zunehmend autonom. Das Spektrum reicht hier vom autonomen Staubsauger über Fußballroboter und autonome Rollstühle bis hin zu Marsrobotern und selbst steuernden Raumsonden.

In diesem Kapitel wird zunächst in eine grundsätzliche Problematik autonomer Agenten eingeführt, die Handlungskontrolle in dynamischen Umgebungen. Danach werden einige bestehende Lösungsversuche und deren Stärken und Schwächen aufgezeigt. Das 2. Kapitel geht auf einen integrierten Ansatz zur Handlungskontrolle, auf sogenannte Verhaltensnetzwerke [Maes, 1989], näher ein. Sie bilden die Grundlage für die in Kapitel 3 eingeführten erweiterten Verhaltensnetzwerke. Diese verbessern die von Maes eingeführten Verhaltensnetzwerke durch Behebung zahlreicher Probleme und erweitern diese um wesentliche Eigenschaften wie dem situationsabhängigen motivationalen Einfluß durch Ziele und der parallelen Ausführung von Verhalten. In Kapitel 4 werden die konkreten Domänen vorgestellt, die für empirische Untersuchungen mit Verhaltensnetzwerken verwendet wurden, und anhand einer allgemeinen Kategorisierung von Domänen eingeordnet. Für die empirischen Untersuchungen wurde zum einen die klassische Blockwelt-Domäne, zum anderen die RoboCup-Umgebung für simulierten Fußball [Noda, 1995] verwendet. Ergebnisse der empirischen Untersuchungen von erweiterten Verhaltensnetzwerken werden in Kapitel 5 vorgestellt. Abschließend werden in Kapitel 6 die Ergebnisse diskutiert und es wird auf Einschränkungen und offene Fragen eingegangen.

1.1 Autonome Agenten

Autonome Agenten sind dadurch charakterisiert, daß sie sich in einer Umgebung befinden und dadurch selbst Teil der Umgebung sind. Sie können diese Umgebung wahrnehmen und durch ihr Handeln die Umgebung verändern (siehe Abb. 1.1). Die Umgebung bestimmt dabei wesentlich die Anforderungen an den Agenten. Zunächst wird daher eine allgemeine Kategorisierung von Domänen erläutert, bevor näher auf die Anforderungen einer Handlungskontrolle von autonomen Agenten eingegangen wird.

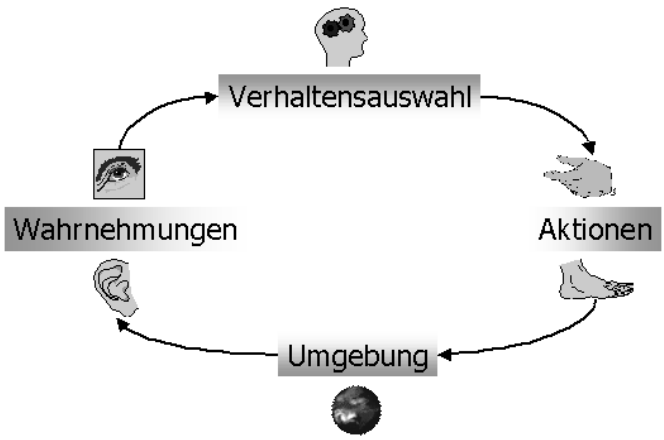

Abbildung 1.1: Agenten nehmen ihre Umgebung wahr, entscheiden sich für auszuführende Verhalten und ändern durch deren Ausführung den Zustand ihrer Umgebung.

1.1.1 Domänen

Ein Agent ist charakteristischer Weise Teil einer Umgebung, der Domäne. Die Domäne ist das Spielfeld, die Welt des Agenten. Die Wahrnehmung liefert dem Agenten Information über den Zustand der Domäne und sich selbst, da er Teil der Domäne ist. Alle Aktionen können den Zustand der Domäne verändern, sind also auf diese beschränkt.

Domänen können entsprechend verschiedener Eigenschaften kategorisiert werden, wobei in der Regel die Eigenschaften der Domäne aus Sicht eines Agenten betrachtet werden [Russel und Norvig, 1995]:

- deterministisch - nicht-deterministisch
  Eine Domäne ist deterministisch, wenn sich aus dem aktuellen Zustand und der Aktion des Agenten der zukünftige Zustand berechnen läßt. Insbesondere bedeutet das, daß alle Aktionen eines Agenten in einer deterministischen Umgebung immer die erwarteten Effekte nach sich ziehen. Dadurch sind solche Umgebungen einfacher als nichtdeterministische, in denen ein Agent lediglich statistische Aussagen über das Eintreffen von Effekten machen kann.

- statisch - dynamisch
  Als statisch wird eine Domäne bezeichnet, wenn ausschließlich Aktionen des Agenten den Zustand der Domäne beeinflussen können. Ändert sich der Zustand ohne Eingriff des Agenten, was insbesondere auch in Multiagentensystemen der Fall ist, spricht man von einer dynamischen Umgebung. Statische Umgebungen sind aus Sicht der Handlungskontrolle einfacher als dynamische, da dem Agenten in einer statischen Umgebung beliebig viel Zeit für die Auswahl der nächsten Aktion zur Verfügung steht. Zudem spielt nur die Reihenfolge der Aktionsausführung eine Rolle, während in dynamischen Umgebungen auch der Zeitpunkt einer Aktion von Bedeutung ist.

- diskret - kontinuierlich
  Die Unterscheidung diskret versus kontinuierlich differenziert Domänen nach der Anzahl der sie beschreibenden Zustände. Gibt es in einer Domäne nur eine beschränkte Anzahl von Zuständen, so ist sie diskret. Sind Zustände hingegen durch kontinuierliche Größen festgelegt, wie z.B. Länge, Gewicht, Geschwindigkeit, handelt es sich um eine kontinuierliche Umgebung.

- zugänglich - nicht zugänglich
  Kann der Agent den gesamten momentanen Zustand der Umgebung wahrnehmen, d.h. besitzt der Agent vollständiges Weltwissen, wird die Domäne als zugänglich bezeichnet. Hat der Agent nur partiellen Einblick in den Zustand der Umgebung, wie es etwa beim eingeschränkten Sehfeld des Menschen der Fall ist, wird sie als nicht zugänglich oder auch als teilweise zugänglich oder teilweise beobachtbar bezeichnet. Zugängliche Umgebungen vereinfachen die Handlungskontrolle, da der Agent kein Gedächtnis für den Zustand der nicht sichtbaren Umgebung benötigt.

Agenten älterer KI-Systeme beschränkten sich fast ausschließlich auf diskrete und deterministische Domänen. Beschränkte Rechenkapazitäten und die prinzipielle Komplexität der optimalen Aktionsauswahl machten dies notwendig. Viele dieser Domänen werden inzwischen aber vom Computer gleich gut oder besser beherrscht als von menschlichen Experten (z.B. Dame, Schach). Neue Herausforderungen werden daher zunehmend in schwierigeren, d.h. nicht deterministischen, dynamischen oder kontinuierlichen Domänen gesucht.

1.1.2 Handlungskontrolle

Ein Grundproblem, das allen Agenten gemeinsam ist, ist das Problem der Handlungskontrolle. Ein Agent muß sich aufgrund der wahrgenommenen Situation, seinem Wissen über die Vergangenheit und seinen Zielen in der Zukunft entscheiden, welche der meist zahlreichen möglichen Aktionen er ausführen will. Die Entscheidung soll also sowohl den langfristigen Nutzen des Verhaltens eines Agenten optimieren als auch der aktuellen Situation angemessen sein.

An dieser Stelle scheint es angebracht, die Begriffe Handlung, Verhalten und Aktion voneinander abzugrenzen, insbesondere deshalb, weil deren Verwendung

etwa in der Psychologie und in der künstlichen Intelligenz-Forschung (KI) teilweise voneinander abweicht.

In der Psychologie wird ein (menschliches) Verhalten als Handlung bezeichnet, wenn es aus Sicht des Handelnden zielgerichtet ist, also einen Sinn hat. Ein Beobachter wird das Verhalten als Handlung bezeichnen, wenn er diesen Sinn erkennen und verstehen kann [Heckhausen, 1989]. "In den psychologischen Handlungstheorien (...) sind neben der Zielgerichtetheit die Bewußtheit, die Situationsbezogenheit, die sequentielle sowie die hierarchische Organisation des Handelns wesentliche Bestandteile einer Definition, die Handeln beispielsweise vom Begriff des Verhaltens abgrenzt." [Strube, 1996]. Als Verhalten werden im behavioristischen Sinn alle beobachtbaren Reaktionen eines Organismus auf einen Reiz bezeichnet. Im Unterschied zu Handlungen kann man Verhalten zwar Ursachen, aber keinen Sinnzusammenhang zuweisen [Heckhausen, 1989]. So kann dasselbe Verhalten unterschiedlichen Handlungen zugrundeliegen. Zum Beispiel kann das Verhalten, den Telefonhörer abzunehmen Teil der Handlung sein, ein Gespräch entgegenzunehmen. Es kann aber auch mit der Intention geschehen, das Telefon zu blockieren, damit keine Anrufe mehr durchkommen.

In der KI wurden zunächst alle Interaktionen eines Agenten mit der Umgebung als Aktion (action) bezeichnet. Ausgehend von diskreten und statischen Umgebungen ist eine Aktion formal die von einem Agenten verursachte Überführung eines Zustands in einen Folgezustand. Aktionen stehen also für einzelne diskrete Interaktionen mit der Umgebung. Für zielgerichtetes Handeln werden Aktionen z.B. in Handlungsplänen zusammengefaßt. Mitte der achtziger Jahre wurde zunehmend auch der Begriff des Verhaltens (behavior) für fest kodierte, situationsgesteuerte Interaktionen mit der Umwelt verwendet [Brooks, 1986]. Im Unterschied zur ausschließlich phenomenologischen Verwendung des Begriffs im Behaviorismus verwenden Brooks u.a. den Begriff Verhalten zur Bezeichnung der Bausteine für die Interaktion mit der Umgebung.

Im Rahmen dieser Arbeit wird der Begriff *Handlung* für zielgerichtetes, aber nicht notwendigerweise bewußtes Verhalten verwendet. Beispiel für eine Handlung im Fußball ist, das Verhalten in Richtung gegnerisches Tor zu rennen mit der Absicht auszuführen, den Gegner ins Abseits zu stellen. Als *Verhalten* werden im Sinne der Brooks'schen Verwendung des Begriffs zeitlich ausgedehnte, situationsgesteuerte Aktionssequenzen bezeichnet. Verhalten werden also durch die Ausführung einzelner Aktionen realisiert. Dasselbe Verhalten kann dabei Bestandteil unterschiedlicher Handlungen sein. *Aktionen* bilden die kleinste Einheit möglicher Interaktion mit der Umgebung und sind von kurzer zeitlichicher Dauer. Wiederum kann dieselbe Aktion Bestandteil unterschiedlicher Verhalten sein.

1.1.3 Anforderungen an Handlungskontrolle

Im folgenden sind einige Kriterien aufgeführt, die ein Agent in einer dynamischen, teilweise zugänglichen und nicht-deterministischen Umgebung erfüllen muß, um erfolgreich zu sein. Je mehr dieser Eigenschaften ein Agent auf sich vereinigt, desto besser sollte er in einer solchen Umgebung zurecht kommen.

1. Der Nutzen des Verhaltens wird zumeist an den Zielen des Agenten, bzw. am Beitrag des Verhaltens zu deren Erreichung gemessen. Wichtig ist, daß komplexe Agenten mehrere auch in Konflikt zueinander stehende Ziele gleichzeitig haben können. Diese Ziele sind in der Regel unterschiedlich wichtig und je nach Situation unterschiedlich relevant. Handlungskontrolle sollte also in der Lage sein, eine Art von Zielmanagement zu betreiben, d.h. mehrere Ziele bei der Verhaltensauswahl zu berücksichtigen und gegebenenfalls Zielkonflikte zu lösen. Dazu gehört auch, bei der Verfolgung eines Ziels günstige Gelegenheiten zum Erreichen anderer Ziele wahrzunehmen. Können also durch einen geringn Umweg weitere Ziele erreicht werden, sollte der Agent dies erkennen.

2. Der Situation angemessen ist eine Handlungskontrolle, wenn sie auf Änderungen in der Umgebung, wie z.B. auf eine Gefahrensituation, ausreichend schnell reagiert. Dies ist gerade in dynamischen Umgebungen notwendig, in denen sich der Zustand der Umgebung auch ohne das Zutun des Agenten, also unvorhersehbar, verändern kann. Dazu ist es notwendig, daß sowohl die Entscheidung für ein Verhalten als auch dessen Ausführung effizient ist.

3. Es kann von Vorteil oder auch zwingend notwendig sein, mehrere Verhalten gleichzeitig auszuführen, bzw. verschiedene Verhalten durch Übung in ein Verhalten zu verschmelzen. Beispielsweise ist es hilfreich, bei der Flucht vor einer Gefahr gleichzeitig auch andere vor dieser Gefahr zu warnen. Für das Fahren eines Autos ist es notwendig, gleichzeitig zu kuppeln und den Gang einzulegen. Es ist zu erwarten, daß eine Handlungskontrolle, die es erlaubt, mehrere Verhalten gleichzeitig auszuführen, in dynamischen Umgebungen erfolgreicher ist, da das Handeln in solchen Umgebungen zeitkritisch ist. Mehrere Verhalten gleichzeitig auszuführen bedeutet, ein Ziel schneller zu erreichen, bzw. mehrere Ziele gleichzeitig zu verfolgen.

4. Handlungskontrolle in realen Umgebungen sollte außerdem robust sein. Zum einen ist in solchen Umgebungen meist keine perfekte Wahrnehmung zu erwarten. Das heißt, daß einerseits nicht alle relevanten Informationen über die Umgebung vorliegen müssen und andererseits vorliegende Informationen falsch sein können. Zum anderen kann auch die Ausführung von Aktionen nicht als deterministisch angenommen werden. Die Effekte einer Handlung treten allenfalls mit einer gewissen Wahrscheinlichkeit auf, was sowohl an Ungenauigkeiten der Aktorik des Agenten als auch am Nichtdeterminismus der Umgebung selbst liegen kann. Robust ist eine Handlungskontrolle dann, wenn sie bei solchen Problemen nicht völlig zusammenbricht, sondern allenfalls mit zunehmenden Problemen graduell in der Qualität abnimmt.

5. Damit sich die Qualität der Handlungskontrolle mit der Zeit verbessert und auf langfristige Änderungen in der Umgebung angemessen reagiert, sollte sie adaptiv sein, d.h. sie sollte aus Fehlern und Erfolgen der Ver-

gangenheit für die Zukunft lernen. Idealerweise kann so das komplette Domänenwissen durch Exploration der Umgebung gelernt werden. Oft ist es jedoch nicht sinnvoll oder effektiv, komplettes Domänenwissen erlernen zu lassen. In solchen Fällen sollte der Mechanismus der Handlungskontrolle es möglichst einfach erlauben, Domänenwissen vorzugeben, zu erweitern oder zu ändern.

6. Schließlich ist es in Multiagentensystemen notwendig, das eigene Handeln auf die Anforderungen und Handlungen anderer Agenten abzustimmen. Idealerweise handeln die Agenten nach einem gemeinsamen Plan und sind so erfolgreicher, als es ein einzelner Agent sein kann. Beispiele hierfür sind die Jagd im Rudel oder das Fußballspiel.

Die beiden Anforderung an Handlungskontrolle, den langfristigen Nutzen zu optimieren und, wenn nötig, schnell zu agieren, bilden die beiden Pole der Handlungskontrolle autonomer Agenten. Eine rein deliberative Handlungskontrolle optimiert den langfristigen Nutzen und berücksichtigt ausschließlich die Zielerreichung des Agenten. Eine rein reaktive Verhaltenskontrolle berücksichtigt ausschließlich den momentanen Zustand für eine schnelle Verhaltensauswahl.

Im folgenden wird zunächst auf diese beiden Paradigmen der Handlungskontrolle von Agenten eingegangen. Danach werden zwei Kategorien von Ansätzen vorgestellt, die versuchen, diese beiden Anforderungen an intelligente Handlungskontrolle in sich zu vereinigen. Das sind zum einen die hybriden Architekturen, die jeweils eine reaktive und eine deliberative Komponente besitzen und zum anderen integrierte Ansätze.

1.2 Deliberative Handlungskontrolle

Die Basis deliberativer Handlungskontrolle bildet der Begriff der rationalen Entscheidung. Eine Entscheidung wird als rational bezeichnet, wenn sie auf die Erreichung von Zielen ausgerichtet ist. "Acting rationally means acting so as to achieve one's goals, given one's believes." [Russel und Norvig, 1995]. Als Grundlage für rationale Entscheidungen in nicht-deterministischen Umgebungen gilt das Erwartungs-mal-Wert Prinzip der Entscheidungstheorie. Es besagt, daß eine Entscheidung sowohl den Wert als auch die Unsicherheit der Konsequenzen berücksichtigen soll.

Die Entscheidungstheorie hat ihre Wurzeln in der Mathematik und in der Ökonomie und spielt sowohl in der Psychologie als auch in der künstlichen Intelligenz eine Rolle. Pascal rät bereits im 17. Jahrhundert, beim Glücksspiel auf die Variante zu setzen, deren Produkt aus Gewinn und Eintretenswahrscheinlichkeit maximal ist. Im 18. Jahrhundert entwickelten Bernoulli und Laplace die Grundlagen der Wahrscheinlichkeitstheorie, die in Verbindung mit der Nutzentheorie die Basis für rationale Entscheidungen nach dem Erwartungs-Wert Prinzip bildet.

In der Ökonomie verwendeten Smith im 18. und Mill im 19. Jahrhundert entscheidungstheoretische Ansätze. Von Neumann und Morgenstern [1944] legten erstmals eine umfassende Theorie präferentieller Entscheidung vor. Vier

Arbeiten von Wirtschaftswissenschaftlern auf dem Gebiet der Entscheidungs- und Spieltheorie wurden mit Nobelpreisen ausgezeichnet (Simon 1978, Allais 1988, Becker 1992, Selten 1994) [Jungermann *et al.*, 1998].

In der Psychologie wurde versucht, tatsächliche Entscheidungen mittels Entscheidungstheorie vorauszusagen. Doch bereits Bernoulli wies 1738 darauf hin, daß Menschen statt des zu erwartenden objektiven Werts einen subjektiven Wert der Entscheidung zugrunde legen. Weiterhin stellte man fest, daß auch die subjektive Wahrscheinlichkeit von der objektiven Wahrscheinlichkeit abweichen kann. Besonders hohe Wahrscheinlichkeiten werden überschätzt, niedrige Wahrscheinlichkeiten werden unterschätzt. Dies führte zur Einführung des subjektiv erwarteten Nutzens. Dieser kann nach Lewin, Dembo, Festinger und Sears [1944] als gewichtete Summe der positiven Valenz V_e gewichtet mit deren (subjektiven) Eintrittswahrscheinlichkeit W_e und der negativen Valenz V_m gewichtet mit deren Eintrittswahrscheinlichkeit W_m der Effekte einer Handlung berechnet werden: $V = V_e \cdot W_e + V_m \cdot W_m$. Die Theorie sagt voraus, daß eine Entscheidung auf die Alternative fällt, deren gewichtete Summe aus Erfolgs- und Mißerfolgsvalenz maximal ist. Atkinson [1957] erweiterte den Begriff der Valenz indem er ihn nach Anreiz A und persönlicher Motivstärke M differenzierte: $T = M_E \cdot A_e \cdot W_e + M_M \cdot A_m \cdot W_m$. Zudem sah er die Erfolgs- und Mißerfolgswahrscheinlichkeiten als komplementär an ($W_e = 1 - W_m$) und stellte den Anreiz in einen linearen Zusammenhang mit der subjektiven Erfolgswahrscheinlichkeit ($A_e = 1 - W_e$, $A_m = 1 - W_m$). Diese Arbeit beeinflußte die Motivationspsychologie in den folgenden 20 Jahren nachhaltig [Strube, 1998].

Auch in der KI spielt die Entscheidungstheorie als Grundlage für Entscheidungen in nicht-deterministischen Umgebungen eine wesentliche Rolle. Sie kommt unter anderem in Entscheidungsnetzwerken [Howard und Matheson, 1984] und Expertensystemen [Horvitz *et al.*, 1988] sowie beim Berechnen optimaler Policies bei Markov Entscheidungsproblemen zum Einsatz (siehe auch Kapitel 1.5.2). Zunächst stand in der KI aber deliberative Handlungskontrolle und Handlungsplanung in deterministischen Umgebungen im Vordergrund.

1.2.1 Künstliche Intelligenz

Deliberative Handlungskontrolle ist in der künstlichen Intelligenz durch die explizite Repräsentation des Zustandsraums und das Schlußfolgern innerhalb der Repräsentation gekennzeichnet. Mittels Wissen über Effekte von Handlungen werden zukünftige Situationen antizipiert, wodurch Handlungen bevorzugt werden können, die zu Situationen führen, in denen Ziele des Agenten erfüllt sind.

Wichtige Kriterien in der künstlichen Intelligenz Forschung sind die Korrektheit, Vollständigkeit und möglichst auch die Optimalität der verwendeten Algorithmen, sowie deren Ausdrucksfähigkeit und komplexitätstheoretischen Eigenschaften.

Handlungsplanung

Zielgerichtetes Handeln wird in der klassischen Handlungsplanung dadurch erreicht, daß unter Benutzung des Wissens über die Effekte von Aktionen Akti-

onsketten gebildet werden, die, ausgehend vom angegebenen Startzustand, die Erreichung der Ziele gewährleisten. Als erstes Handlungsplanungssystem wird zumeist STRIPS [Fikes und Nilsson, 1971] angeführt, dessen Planungsalgorithmus als Suche im Zustandsraum aufgefaßt werden kann. Nichtlineares Planen [Sacerdoti, 1974; 1977], [McAllester und Rosenblitt, 1991] führt dagegen eine Suche im Raum der möglichen Pläne durch. Die Operatoren eines nichtlinearen Plans sind nur teilweise geordnet, eine endgültige Festlegung der Reihenfolge von Aktionen erfolgt erst bei der Ausführung des Plans. Ein nichtlinearer Plan kann also mehrere lineare Pläne repräsentieren. Durch die späte Festlegung der Reihenfolge der Operatoren (least commitment) wird der Suchraum eingeschränkt. Mit UCPOP [Penberthy und Weld, 1992] wurde die Ausdrucksfähigkeit nichtlinearer Planer z.B. um bedingte Effekte und allquantifizierte Vorbedingungen erweitert.

Annahmen der klassischen Handlungsplanung sind, daß die betrachtete Umgebung statisch, deterministisch und vollständig zugänglich ist. Die Ziele sind konsistent und explizit vorgegeben, und für die Erstellung des Plans ist beliebig viel Zeit vorhanden. Moderne Planungsverfahren versuchen, diese Einschränkungen abzuschwächen oder aufzuheben. Hierarchisches Planen [Erol *et al.*, 1994; Sacerdoti, 1977] erstellt zunächst Pläne unter Verwendung abstrakter Operatoren und versucht dann diese zu verfeinern, wodurch der Suchraum stark eingeschränkt werden kann. Schnellere Planungsverfahren (z.B. Graphplan [Blum und Furst, 1997]) werden entwickelt, um größere oder zeitbeschränkte Planungsaufgaben lösen zu können. Probabilistisches Planen (z.B. [Kushmerick *et al.*, 1995]) erweitert den Anwendungsbereich auf nicht-deterministische Umgebungen. Konditionales Planen sowie teilweise zugängliche Markov Entscheidungs Prozesse (POMDPs) [Cassandra *et al.*, 1994] erweitern den Anwendungsbereich auch auf nicht vollständig zugängliche Umgebungen (siehe Kapitel 1.5.2).

In dynamischen Umgebungen erweist sich Handlungsplanung dennoch oft als nicht ausreichend oder sinnvoll. Durch die schnellen und unvorhergesehenen Änderungen der Situation in solchen Umgebungen verlieren Pläne schnell ihre Gültigkeit. Je dynamischer eine Umgebung ist, desto weniger lohnt sich der Aufwand, langfristig zu planen. Zudem ist die Komplexität vieler Planungsprobleme (siehe z.B. [Chapman, 1987; Bylander, 1994; Littman *et al.*, 1995]) für die Anwendung in größeren Domänen prohibitiv.

Belief Desire Intention

Motivationale Aspekte von Handlungskontrolle sowie Elemente rationaler Entscheidung werden in der künstlichen Intelligenz unter anderem im Rahmen von sogenannten BDI-Architekturen behandelt (Belief - informationaler Zustand, Annahme; Desire - motivationaler Zustand, Ziel; Intention - deliberativer Zustand des Agenten, intendiertes Ziel). Basierend auf den Arbeiten von Bratman [1987; 1990] und Cohen und Levesque [1987] stellten Rao und Georgeff [1991] eine formale Theorie für eine BDI-Architektur vor. Die Umgebung wird darin als Zustandsbaum repräsentiert mit Verzweigungen für mögliche Zustände in der Zukunft und einer Vergangenheitslinie. Annahmen des Agenten sind dabei

die für möglich gehaltenen Zustände, Ziele sind wünschenswerte Zustände, und Intentionen sind gerade angestrebte wünschenswerte Zustände der Umgebung. Verzweigungspunkte des Zustandsbaums repräsentieren Wahlmöglichkeiten des Agenten (choice nodes) sowie Nichtdeterminismus von Aktionen des Agenten (chance nodes).

Basierend auf diesen theoretischen Konzepten wurden konkrete Implementierungen vorgestellt [Fischer *et al.*, 1995], [Rao und Georgeff, 1995], [Burkhard *et al.*, 1998]. Die beiden letztgenannten fügen zur Auswahl von Intentionen aus der Menge der Ziele den Verzweigungspunkten des Zustandsbaums Information über die geschätzte Wahrscheinlichkeit des Eintreffens (chance nodes) sowie Information über den Nutzen eines Zustands (choice nodes) hinzu. Pfade im Zustandsbaum mit entsprechend hohem erwartetem Nutzen werden bevorzugt als neue Intention gewählt.

BDI-Architekturen erlauben die Berücksichtigung motivationaler Zustände des Agenten. Ein Agent kann mehrere, unterschiedlich wichtige Ziele besitzen, von denen er zu einem Zeitpunkt nicht alle anstrebt. Ein weiterer Vorteil ist, daß Nichtdeterminismus der Aktionen des Agenten explizit repräsentiert werden kann. Problematisch erscheint jedoch, daß die Handlungskontrolle des Agenten ausschließlich zielgerichtet ist. Reaktionen auf unerwartete Situationen sind nur durch die Bildung entsprechender neuer Ziele und Intentionen, also durch einen kompletten Deliberationsprozeß möglich. Das kann in dynamischen Umgebungen zu lange dauern, wenn beispielsweise auf eine Gefahrensituation reagiert werden soll (siehe auch [Strube, 1998]).

1.2.2 Kognitionswissenschaft

Im Zentrum kognitionswissenschaftlicher Forschung steht die Erforschung und Modellierung menschlicher Informationsverarbeitung. Dabei stand zunächst das Paradigma der Symbolverarbeitung (physical symbol system hypothesis [Newell und Simon, 1976]) im Vordergrund. Neben der Erforschung und Modellierung vieler Einzelphänomene wurde auch versucht, mit Hilfe umfassender kognitiver Architekturen ein breites Spektrum menschlicher Kognition zu modellieren. Im folgenden sind die beiden wohl bekanntesten kognitiven Architekturen Soar und Act-R vorgestellt. Einen detaillierten Vergleich liefert Johnson [1997]

Soar

Soar (State, Operator, Apply and Result) [Laird *et al.*, 1987; Rosenbloom *et al.*, 1993] ist eine kognitive Architektur mit dem Anspruch, eine vereinheitlichte Theorie der Kognition zu modellieren [Newell, 1990]. Eine wesentliche Richtlinie beim Entwurf von Soar war es, die Anzahl der verwendeten Mechanismen zu minimieren. So verfügt Soar über genau einen Mechanismus zur Langzeitspeicherung von Wissen, den Produktionen, genau eine Repräsentation für das Arbeitsgedächtnis nämlich Objekte mit Attributen und zugehörigen Werten, einen Mechanismus zur Generierung von Unterzielen (subgoaling) und einen Lernmechanismus, das Zusammenfassen von Produktionen (chunking). Der Entscheidungszyklus verläuft in zwei Phasen. In der Elaborationsphase

werden alle Produktionen parallel ausgewertet und die, die dem aktuellen Kontext entsprechen, parallel gefeuert. Das feuern einer Produktion kann etweder Objekte und Werte der zentralen Datenstruktur im Arbeitsgedächtnis, dem Kontextstapel (context stack), hinzufügen bzw. ändern, Operatoren (Aktionen) vorschlagen oder Präferenzen für bereits vorgeschlagenen Operatoren vergeben. In der Entscheidungsphase wird dann der Operator ausgeführt, der die höchste Präferenz hat. Liegt zu wenig oder widersprüchliche Information für die Entscheidung vor, tritt eine Stocksituation (impasse) ein, und ein neues Unterziel wird gebildet und auf den Zielstapel gelegt. Anders als bei Act-R werden aber alle Ziele im Zielstapel, angefangen vom top-level Ziel, von den Operatoren berücksichtigt. Dadurch kann Soar auch auf unvorhergesehene Situationen reagieren, solange sie Ziele im Zielstapel betreffen. Durch die Beschränkung auf eine Zielhierarchie ist es in Soar aber schwierig, parallele Handlungen auszuführen, die nicht derselben Zielhierarchie zuzuordnen sind [Tambe *et al.*, 1995]. Werden Ziele unterschiedlicher Hierarchieen auf den Zielstapel gelegt, können bei Erreichung eines Ziels durch das Entfernen aller Unterziele in Soar auch Ziele der eigentlich unabhängigen zweiten Zielhierarchie gelöscht werden, da sie nicht als eigene Zielhierarchie erkannt werden können [Tambe *et al.*, 1995]. Ein weiterer Nachteil von Soar ist, daß es weder die Zuverlässigkeit von Operatoren noch unterschiedliche Wichtigkeit von Zielen bei der Auswahl von Operatoren berücksichtigt [Johnson, 1997].

Act-R

Ähnlich wie Soar ist Act-R [Anderson, 1993; Anderson und Lebiere, 1998] eine kognitive Architektur, die auf dem Prinzip der Produktionensyteme beruht. Anders als bei Soar ist die Trennung von Langzeit- und Kurzzeitgedächtnis nicht explizit durch unterschiedliche Repräsentationsformen gegeben. In Act-R wird vielmehr zwischen prozeduralem (productions) und deklarativem (working memory elements WMEs) Gedächtnis unterschieden. WMEs sind (implizit) Teil des Arbeitsgedächtnisses, wenn sie eine hohe Aktivierung besitzen, welche sie durch (häufige) Benutzung und durch assoziative Verbindungen erhalten. Das Verhalten in Act-R ist immer zielgerichtet. Die Ziele und Unterziele werden durch Produktionen gebildet und in einem Zielstapel verwaltet. Immer das oberste Ziel im Stapel ist dabei aktiv. Produktionen werden parallel instanziiert, aber nur eine Produktion darf zu einem Zeitpunkt feuern und nur dann, wenn sie das aktuelle Ziel unterstützt. Welche Produktion feuern darf wird aus der erwarteten Wahrscheinlichkeit, mit der das Ziel bei Ausführung der Produktion erreicht wird und dem erwarteten Nutzen des Ziels berechnet. Übersteigen die erwarteten Kosten für weitere Instanziierungen den erwarteten Mehrgewinn gegenüber der bisher besten Produktion, wird diese gefeuert, da sie gut genug (satisficing [Simon, 1978]) ist.

Das Handeln in Act-R ist immer zielgerichtet. Der Kontextstapel bzw. Zielstapel wird zunächst mit dem aktuellen Ziel belegt. Kann dieses nicht direkt erreicht werden, werden Unterziele gebildet. Es entsteht eine Zielhierarchie, wobei aber jeweils nur ein Ziel aktiv sein kann. Dadurch ist Act-R nicht in der Lage, mehrere Ziele gleichzeitig bei der Auswahl von Produktionen zu berück-

sichtigen [Johnson, 1997; Westermann und Heise, 1996]. So können weder Produktionen bevorzugt werden, die mehreren Zielen gleichzeitig zuträglich sind, noch können Gelegenheiten in der Umgebung wahrgenommen werden, mit einem kleinen Umweg ein weiteres Ziel zu erreichen.

Da das Handeln immer zielgerichtet ist, und es immer genau ein aktives Ziel gibt, kann auch auf unvorhergesehene Änderungen in der Situation nicht schnell reagiert werden. Generell liegt der Schwerpunkt von Act-R eher im Problemlösen als in der Handlungskontrolle in dynamischen Umgebungen. Was fehlt, ist die Situiertheit, die Einbindung in die Umgebung. Dem wird in der neuen Version Act-R/PM [Anderson und Lebiere, 1998] durch die Einführung eigenständiger und parallel arbeitender Sensor- und Aktormodule entgegengewirkt. Nach wie vor bleibt aber die Beschränkung des einen Zielstapels mit einem aktiven Ziel.

1.3 Reaktive Verhaltensskontrolle

Im Gegensatz zur deliberativen Handlungskontrolle, deren Schwerpunkt die formale Repräsentation des Zustandsraums und das Schlußfolgern innerhalb der Repräsentation bildet, ist es das Paradigma der reaktiven[1] Verhaltenskontrolle, ganz oder überwiegend ohne explizite Repräsentation auszukommen. Die Umgebung selbst soll als Repräsentation verwendet werden: 'The world is its own best model' [Brooks, 1991]. Reaktive Verhaltenskontrolle besteht also darin, eine Abbildung $f : S \rightarrow A$ von der momentanen Situation auf eine Aktion zu finden. Paradigmen reaktiver Verhaltenskontrolle sind nicht Optimalität und Vollständigkeit, sondern Robustheit und Schnelligkeit der verwendeten Algorithmen. Insbesondere soll der Problematik Rechnung getragen werden, daß Informationen über echte und dynamische Umgebungen immer unvollständig und oft unkorrekt sind.

1.3.1 Biologische Systeme

Bereits einfache Organismen reagieren auf Reize aus der Umgebung. So ziehen beispielsweise Amöben ihre Pseudopodien (Scheinfüßchen) zurück, wenn man sie mit einer Nadel sticht. Auch bei kognitiven Systemen, wie dem Menschen, existieren neben den kognitiven Prozessen auch ausschließlich reizgesteuerte Mechanismen. Bekannt ist der Patellarsehnenreflex, der bei einem Schlag gegen die Patellarsehne unterhalb der Kniescheibe eine Muskelkontraktion im Oberschenkel auslöst. Aber auch die meisten Köperfunktionen des Menschen sind datengesteuert und kognitiv nicht penetrabel, d.h. sie können weder beeinflußt noch unterdrückt werden [Strube, 1998]. Tritt der auslösende Reiz z.B. eines motorischen Reflexes auf, wird unwillkürlich der entsprechende Reflex ausgeführt. Der Vorteil ist, daß Reflexe sehr schnell ausgeführt werden können, da Sensorik und Aktorik direkt miteinander verbunden sind. Nachteilig ist, daß eine solche direkte Kopplung sehr starr ist.

[1] Anstatt von reaktiver wird oft auch von situierter oder verhaltensbasierter Verhaltenskontrolle gesprochen.

Höhere Organismen zeichnen sich dadurch aus, daß sie lernfähig sind, d.h. ,daß Sensorik und Aktorik nur indirekt miteinander verbunden sind. Durch klassisches Konditionieren können z.B. reflexhafte Vorgänge mit Reizen assoziert werden, was bereits bei Organismen wie dem Regenwurm gelingt [Gallistel, 1990]. Menschen sind schließlich zu kognitiven Leistungen wie Planung in der Lage. Bemerkenswert ist jedoch, daß Reflexe nicht von kognitiven Prozessen abgelöst wurden, sondern mit ihnen in einem Organismus koexistieren [Strube, 1998].

1.3.2 Künstliche Intelligenz

Reaktive Verhaltenskontrolle wurde in der künstlichen Intelligenz Forschung verstärkt Mitte der achtziger Jahre propagiert. Verschiedene Architekturen wurden vorgeschlagen, um möglichst einfach robustes Verhalten zu erzielen. Schwerpunkt war dabei die Arbeit mit echten Robotern. Im folgenden sind einige dieser Architekturen vorgestellt. In späteren Arbeiten wurde zunehmend versucht, reaktives und zielgerichtetes Handeln zu kombinieren, da es sich bei rein reaktiven Ansätzen als schwierig erwies, ein gewünschtes zielgerichtetes Verhalten zu realisieren.

Subsumptionsarchitektur

Vorreiter für diese Position in der KI-Forschung war Brooks Subsumptionsarchitektur [Brooks, 1986]. Wesentliches Merkmal ist der horizontale Schichtenaufbau von immer komplexeren Verhalten, die jeweils direkt mit der Sensorik und Aktorik verbunden sind und die darunter liegenden Verhalten subsumieren, bzw. hemmen oder unterdrücken können. Dadurch wird eine sehr enge Kopplung von Sensorik und Aktorik gewährleistet. Gleichzeitig bleiben beim Hinzufügen von komplexeren Verhalten bisher bestehende Verhalten unverändert. Durch schrittweises Hinzufügen immer komplexerer Verhalten sollten dann höhere kognitive Fähigkeiten wie die Erstellung von Karten oder Handlungsplanung erreicht werden [Brooks, 1986].

Mit Hilfe der Subsumptionsarchitektur gelang es, echte Roboter relativ schnell mit einfachen aber robusten Verhalten wie der Hindernisvermeidung oder Wander- und Explorierverhalten auszustatten [Brooks, 1986; 1989]. Das Ziel auch höhere kognitive Fähigkeiten zu modellieren wurde aber nach meinem Wissen nie erreicht. Insbesondere ist es nicht möglich, Ziele explizit zu repräsentieren. Entsprechend schwierig ist es, zielgerichtetes Verhalten der Agenten zu erreichen.

Pengi

Pengi [Agre und Chapman, 1987] ist ein simuliertes Computerspiel, dessen zentraler Akteur vom Computer gesteuert wird. Der Agentenarchitektur liegt die Feststellung zugrunde, daß der größte Teil der Aktivitäten nicht geplant werden muß, sondern routinemäßig abgearbeitet werden kann (siehe auch [Agre und Chapman, 1990]). Dementsprechend genügt für solche Aufgaben ein einfacher Kontrollmechanismus zur Steuerung des Agenten. Er besteht bei Pengi

aus peripheren Systemen, die für die Sensorik und Aktorik zuständig sind und aus einem zentralen System, das die Kontrolle des Agenten übernimmt und aus kombinatorischen Netzwerken besteht.

Erwähnenswert ist die in Pengi verwendete deiktische Repräsentation der Objekte des Pengo Spiels. Diese auch als indexical-functional aspects bezeichnete Repräsentation benennt Objekte nicht absolut, sondern relativ zu ihrer Beziehung und Funktion zum Agenten. Ein Objekt wird beispielsweise nicht als 'Objekt018' sondern als 'Objekt-vor-mir' bezeichnet. Dadurch kann die Zahl der zu repräsentierenden Objekte deutlich verringert werden, wenn die Anzahl gleichartiger Objekte deutlich größer ist, als die Zahl der in einer Situation relevanten Objekte, was in vielen Umgebungen der Fall ist. Beispielsweise ist in der in dieser Arbeit verwendeten Fußballumgebung die Zahl der in einer Situation relevanten Spieler selten größer als fünf, während die absolute Zahl der Spieler 22 beträgt.

Während die Zuweisung von funktionalen Rollen an Objekte der Umgebung (deictic reference grounding) in Pengi noch handkodiert war, stellten Schoppers und Shu [1996] einen Mechanismus vor, der dieses Problem auf der Symbolebene behandelt und dem Agenten erlaubt, den Objekten der Umgebung aktiv funktionale Rollen zuzuweisen.

Dual Dynamics

Mit Hilfe von Differentialgleichungen werden in Dual Dynamics [Jäger und Christaller, 1997] die Verhalten sowie die Verhaltensauswahl autonomer Agenten modelliert. Die Differentialgleichungen für Verhalten (target dynamics) geben dabei eine Aktivierungstrajektorie für alle für das Verhalten relevanten Aktuatoren an. Die Differentialgleichungen für die Verhaltensauswahl (activation dynamics) legen die Beziehung zwischen abstrakten und elementaren Verhalten fest. Die Verhaltensauswahl wird dabei nicht als Auswahl eines Verhaltens aus vielen verstanden. Abstrakte Verhalten beeinflussen lediglich mittels Aktivierung elementarere Verhalten und steuern so deren Zustand (mode), bzw. legen die Situationen für die Ausführung des elementareren Verhaltens fest. Neben der formalen Klarheit hat die Verwendung von Differentialgleichungen den Vorteil, daß sie den gesamten zeitlichen Verlauf der Aktivierung, bzw. der Verhaltensausführung repräsentieren und damit potentiell kontinuierlicheres Verhalten produzieren. Ein weiterer Vorteil ist, daß beliebige Hierarchien von Verhalten modelliert werden können, wobei untere Ebenen von Verhalten auch dann noch funktionieren, wenn die darüberliegenden Ebenen entfernt werden. Der Nachteil ist, daß es (zumindest für den ungeübten Benutzer) schwierig erscheint, die für ein zielgerichtetes Verhalten notwendigen Differentialgleichungen aufzustellen zumal Ziele nicht explizit repräsentiert werden können.

1.4 Hybride Architekturen

Eine naheliegende Lösung, reaktive Verhaltenskontrolle und deliberative Handlungskontrolle zu kombinieren ist es, jeweils eine reaktive und eine deliberative Komponente in einer Architektur zu vereinigen. Dazu muß geklärt werden, wie

diese beiden Komponenten zusammenarbeiten. Dieses *Kohärenzproblem* wird auf verschiedene Arten gelöst.

Ein Vorschlag ist, je nach Situation zwischen reaktiver und deliberativer Kontrolle hin und her zu schalten [Jensen und Veloso, 1998]. Solange nichts unvorhergesehenes passiert, wird das Verhalten des Agenten von der deliberativen Kontrolle gesteuert. Tritt eine unerwartete Situation in Form einer günstigen Gelegenheit oder eines Ausführungsfehlers ein, wird auf reaktive Kontrolle umgeschaltet, bis der Planer einen neuen, der Situation angemessenen Plan fertiggestellt hat. Die Entscheidung, wann eine solche Situation vorliegt, ist aber keinesfalls trivial.

Einige hybride Architekturen verwenden für die Lösung des Kohärenzproblems ein zwischen reaktiver und deliberativer Komponente liegendes *vermittelndes Modul* (z.B. *executor* in RAP [Firby, 1989], *sequencer* in ATLANTIS [Gat, 1992; 1997] und 3T [Bonasso *et al.*, 1996]). Der RAP-*executor* ist dafür verantwortlich, daß die von der Planungsebene gelieferten nicht instanziierten Pläne (RAPs) in kontinuierliche Handlungen umgesetzt werden. Handlungskontrolle wird in RAP also von der deliberativen Ebene angestoßen und mittels des *executors* vom *controller* ausgeführt. In ATLANTIS und 3T sorgt der *sequencer* selbst für die Handlungsinitiierung. Er übergibt direkt ausführbare Aufgaben an das reaktive Modul (*skill manager*) und delegiert Planungsaufgaben an das deliberative Modul (*planner*). Damit verwaltet er auch die Rechenzeit, die der reaktiven und deliberativen Komponente zugeteilt wird. Allen drei Architekturen ist gemeinsam, daß sie keine motivationalen Aspekte beinhalten. Der Agent hat weder die Möglichkeit, die unterschiedliche Wichtigkeit von Zielen zu berücksichtigen, noch Information darüber, welche von verschiedenen Alternativen zur Erreichung eines Ziels am erfolgversprechendsten sind.

Andere hybride Architekturen (z.B. Reactive Deliberation [Sahota, 1994], INTERRAP [Müller, 1995; 1996], Xavier [Simmons *et al.*, 1997], RHINO [Burgard *et al.*, 1998]) verwenden als Mittel zur Koordination der reaktiven und deliberativen Komponenten *direkte Kommunikation* zwischen diesen. INTERRAP beispielsweise, das neben einer reaktiven (*behavior based layer*, BBL) und einer deliberativen (*local planning layer*, LPL) noch eine dritte Ebene (*cooperative planning layer*, CPL) mit sozialen Kompetenzen enthält, verwendet Bottom-Up Kommunikation, um Verhalten zu initiieren. Stellt die BBL fest, daß sie nicht kompetent für die aktuelle Situation beim gegenwärtigen Ziel ist, übergibt sie die Kontrolle an die LPL. Entsprechendes gilt für die LPL und die CPL. Top-Down Kommunikation wird verwendet, um Verhalten auszuführen. Die CPL übergibt partielle Pläne, die mit anderen Agenten abgestimmt wurden, an die LPL, wo sie in den lokalen Zeitplan übernommen werden. Zur eigentlichen Ausführung von Verhalten sendet die LPL konkrete Befehle an die BBL.

Der Vorteil von hybriden Architekturen ist, daß der modulare Aufbau den Entwurf und die Erstellung des Agenten vereinfachen. Die Module können idealerweise getrennt voneinander erstellt und zum Teil auch getestet werden. Durch die Trennung ist es auch möglich, das reaktive und deliberative Modul parallel ausführen zu lassen. Ein Nachteil ist, daß die Lösung des Kohärenzproblems nicht trivial ist und unter Umständen nicht unerhebliche Kosten (in Form von Rechenkapazität) verursacht.

1.5 Integrierte Ansätze

Neben dem Versuch zielgerichtetes und reaktives Verhalten in hybriden Architekturen zu verwirklichen wurden auch integrierte 'monolithische' Ansätze vorgeschlagen. Diese verwenden im Unterschied zu hybriden Architekturen keine getrennten Module für deliberative und reaktive Kontrolle, sondern versuchen, mit Hilfe eines einzigen Mechanismus zielgerichtetes und reaktives Verhalten zu erzielen.

1.5.1 Situated Automata

In [Kaelbling, 1986; Kaelbling und Rosenschein, 1990; Rosenschein und Kaelbling, 1994] beschreiben Kaelbling und Rosenschein, wie die deklarative Spezifikation eines Agenten in Form von Zielen und Zielreduktionsregeln in einen digitalen Schaltkreis (situated automata) kompiliert werden kann. Dieser steuert den Agenten indem er die wahrgenommene Situation und den internen Zustand des Agenten direkt auf Aktionen der Effektoren abbildet. Diese Abbildung muß nicht vollständig sein - Situationen, von denen aus die Erreichung des Ziels nicht möglich ist sind z.B. nicht im Programm enthalten. Sie ist außerdem in der Regel nicht eindeutig - führen mehrere Aktionen in einer Situation in Richtung der Erreichung des Ziels, wird eine nichtdeterministisch ausgewählt. Die Kompilierung der Spezifikation eines Agenten geschieht durch Zielreduktion des obersten Ziels mit Hilfe der vom Benutzer angegebenen Zielreduktionsregeln. Ziele sind entweder Ausführungsziele - das Ziel eine Aktion auszuführen, Erhaltungsziele, Erreichungsziele sowie beliebige bool'sche Verknüpfungen dieser. Zielreduktionsregeln erlauben komplexere Ziele auf mehrere einfache Ziele zu reduzieren, sowie bedingte Reduktion durchzuführen. Sind mehrere Reduktionen denkbar, können diese priorisiert werden, so daß beim Fehlschlagen einer Reduktion die nächste erfolgversprechende Reduktion angewendet werden kann.

Situated Automata arbeiten schnell, da Wahrnehmungen direkt mittels endlicher vorwärtsgerichteter Schaltkreise auf die Effektoren des Agenten abgebildet werden. Trotzdem ist das Verhalten zielgerichtet, da die Schaltkreise zur Steuerung des Agenten aus der Zielreduktion des Ziels des Agenten kompiliert wurden. Situated Automata sind aber weder in der Lage Ziele dynamisch, d.h. abhängig von der Situation, zu priorisieren, noch können in nichtdeterministischen Umgebungen Erfolgswahrscheinlichkeiten von Effekten berücksichtigt werden.

1.5.2 Entscheidungstheoretisches Planen

Mit Hilfe von entscheidungstheoretischem Planen können auch in nicht-deterministischen Umgebungen optimale Handlungstrategien berechnet werden. Als zugrundeliegendes Modell für entscheidungstheoretisches Planen werden in der Regel Markov-Entscheidungsprozesse (MDPs) herangezogen. Ein MDP ist definiert durch

- eine endliche Menge S von diskreten Zuständen,
- eine endliche Menge A von Aktionen,

- eine Zustandsübergangsfunktion $T : S \times A \rightarrow \Pi(S)$, die für jeden Zustand und für jede Aktion eine Wahrscheinlichkeitsverteilung über alle Zustände angibt und

- eine Belohnungsfunktion $R : S \rightarrow \mathbb{R}$, sowie eine Kostenfunktion $C : A \times S \rightarrow \mathbb{R}$.

Grundlegende Annahme ist die Gültigkeit der Markov Eigenschaft, die besagt, daß die Information der aktuellen Situation ausreichend ist, um Vorhersagen für die Zukunft zu machen, daß also frühere Zustände keine zusätzliche Information enthalten.

Man unterscheidet drei Modelle für optimales Verhalten in MDPs:

1. Im Modell mit endlichem Horizont wird versucht, den Wert (Belohnung - Kosten) der nächsten k Aktionen zu maximieren. Das hat zur Folge, daß eine Strategie im endlichen Horizont nicht statisch ist, sondern davon abhängt, wie viele Aktionen dem Agenten noch verbleiben.

2. Das Modell mit unendlichem Horizont maximiert den Langzeitnutzen des Agenten in der Annahme, der Agent habe noch unendlich viele Zeitschritte vor sich. Der Nutzen jeder Aktion wird durch einen Faktor abgezinst, wodurch Aktionen in naher Zukunft am stärksten gewichtet werden. Dadurch wird verhindert, daß der Nutzen von unendlich vielen Aktionen unendlich groß wird.

3. Ein Abwandlung des zweiten Kriteriums ist, den durchschnittlichen Nutzen aller in Zukunft durchzuführenden Aktionen zu maximieren.

Anders als bei klassischer Handlungsplanung kann die optimale Handlungsstrategie nicht als Folge von Aktionen beschrieben werden. Zum einen werden die Effekte von Aktionen in MDPs als nicht-deterministisch angenommen. Zum anderen muß eine optimale Strategie berücksichtigen, daß der Agent während der Ausführung der Strategie neue Information über die Umgebung erhält, die die Aktionsauswahl beeinflussen sollte. Eine Strategie besteht daher aus einer Aktion für jeden möglichen Zustand, die optimale Strategie aus der optimalen Aktion für jeden Zustand. Der Agent entscheidet sich je nach aktuellem Zustand für die Aktion, die den höchsten erwarteten Nutzen besitzt. Die Berechnung einer optimalen Strategie kann durch Wert-Iteration [Bellman, 1957] oder Strategie-Iteration [Howard, 1960] in polynomialer Zeit in der Anzahl Zustände und Aktionen durchgeführt werden [Littman *et al.*, 1995]. Problematisch ist jedoch, daß diese Algorithmen alle Zustände explizit repräsentieren müssen, wobei die Zahl der Zustände exponentiell mit der Anzahl Eigenschaften (Propositionen) wächst (curse of dimensionality). Dadurch sind nur Probleme mit sehr wenigen Propositionen effizient lösbar.

Daher wird versucht, Algorithmen zur Berechnung einer optimalen Strategie zu entwickeln, die keine explizite Aufzählung aller Zustände benötigen. Solche Algorithmen machen sich zunutze, daß viele reale Probleme Struktur besitzen, d.h., daß

- Aktionen in vielen Zuständen dieselben und meist nur wenige Effekte haben,
- die Belohnungsfunktion oft für viele Zustände identisch ist,
- die Kostenfunktion oft nicht vom aktuellen Zustand sondern nur von der Aktion abhängt.

Schoppers [1987] zeigte für deterministische Aktionen, wie man optimale Strategien (universelle Pläne) aus STRIPS-ähnlichen Problemrepräsentationen automatisch erstellen und in Entscheidungsbäumen relativ effizient speichern kann. Obwohl universelle Pläne bei propositionalen Planungsproblemen schlechtestenfalls entweder exponentiell wachsenden Platz oder exponentiell wachsende Zugriffszeit für die Pläne benötigen [Jonsson und Bäckström, 1996], (auch [Ginsberg, 1989]), ist die Hoffnung, daß für viele Probleme effiziente universelle Pläne existieren [Schoppers, 1989; Chapman, 1989]. Neben der Beschränkung auf deterministische Effekte haben universelle Pläne den Nachteil, daß für die Erstellung eines universellen Plans die Ziele bekannt und konsistent sein müssen. Konflikte zwischen Zielen sowie unterschiedlich wichtige Ziele können nicht behandelt werden.

Damit die explizite Aufzählung aller Zustände in (nicht-deterministischen) MDPs vermieden werden kann, müssen sowohl Aktionen, die Belohnungs- und Kostenfunktion als auch die Strategie- und Wertfunktion kompakt repräsentiert werden. Zur Repräsentation nicht-deterministischer Aktionen wurden unter anderem zwei-Schritt Bayes Netzwerke (2TBN) [Boutilier und Poole, 1996] vorgeschlagen. Die Tabellen der bedingten Wahrscheinlichkeiten der 2TBNs können dabei als Entscheidungsbäume [Boutilier *et al.*, 1995] oder als Entscheidungsgraphen [Hoey *et al.*, 1999] kompakt repräsentiert werden. Belohnungs- und Kostenfunktion sowie Strategie- und Wertfunktion können ebenfalls als Entscheidungsbäume oder als Entscheidungsgraphen repräsentiert werden [Boutilier *et al.*, 1999]. Die Verwendung solcher Repräsentationen benötigt bestenfalls exponentiell weniger Platz und Zeit als eine vollständige Aufzählung. Besitzt ein Problem wenig Struktur, kann aber durch den zusätzlichen Aufwand der Erstellung von Baumstrukturen die Berechnung einer optimalen Strategie ein Vielfaches der Standardverfahren betragen [Hoey *et al.*, 1999].

Die Berechnung einer optimalen Strategie ist trotz kompakter Repräsentation in der Regel recht zeitaufwendig. Einer der derzeit kompaktesten Algorithmen, SPUDD [Hoey *et al.*, 1999], benötigt für ein Beispielproblem mit 19 Propositionen knapp zwei Minuten zur Berechnung der optimalen Strategie. Daher wird diese Berechnung 'offline' durchgeführt und die optimale Strategiefunktion 'online' zur Abbildung des aktuellen Zustands auf die optimale Aktion genutzt. Der Nachteil ist, daß dabei eine statische Belohnungsfunktion vorausgesetzt wird. Ändert sich die Belohnungsfunktion, muß eine neue optimale Strategie berechnet werden. Dynamische Belohnungsfunktionen können so nicht berücksichtigt werden. Ein weiterer Nachteil dieser Ansätze zu entscheidungstheoretischem Planen ist, daß sie wesentlich auf der Annahme von diskreten Weltzuständen basieren, also kontinuierliche Zustände nicht effizient repräsentieren können [Boutilier *et al.*, 1999].

1.5.3 Verhaltensnetzwerke

Einen kompakten Ansatz zur Integration von situationsangepaßtem und zielgerichtetem Verhalten stellen Verhaltensnetzwerke [Maes, 1989] dar. Diese bilden die Grundlage der in dieser Arbeit vorgestellten erweiterten Verhaltensnetzwerke. Sie werden im folgenden Kapitel detailliert beschrieben.

Kapitel 2

Verhaltensnetzwerke nach Maes

Verhaltensnetzwerke wurden von Maes [1989] vorgeschlagen und in weiteren Arbeiten schrittweise erweitert [Maes, 1990a; 1990b; 1992]. Sie stellen einen Mechanismus zur Verhaltensauswahl von Agenten in dynamischen Umgebungen zur Verfügung, der sowohl situationsangepaßt als auch zielgerichtet ist. Dies wird durch einen Mechanismus von Aktivierung und Hemmung durch Ziele und der aktuellen Situation sowie der Verhaltensregeln untereinander erreicht. Dieser erlaubt es unter anderem, mehrere, auch in Konflikt zueinander stehende Ziele, gleichzeitig für die Verhaltensauswahl zu berücksichtigen. Zur Unterscheidung von späteren Erweiterungen wird dieser Mechanismus im folgenden mit MASM (Maes action selection mechanism [Tyrrell, 1994]) bezeichnet.

Im folgenden Abschnitt wird zunächst der Aufbau von Verhaltensnetzwerken beschrieben gefolgt vom Mechanismus der Aktivierungsausbreitung und dem eigentlichen Algorithmus zur Verhaltensauswahl. Danach werden einige von Maes selbst vorgestellte Erweiterungen des Basisalgorithmus erläutert und schließlich auf Probleme von MASM-Netzwerken eingegangen.

2.1 Aufbau

Ein Verhaltensnetzwerk [Maes, 1989; 1990a] besteht aus einem Paar $(\mathcal{K}, \Pi)$, wobei

- $\mathcal{K}$ eine Menge sogenannter *Kompetenzmodule* und
- Π eine Menge von *Parametern* zur Steuerung des Netzwerks sind.

Kompetenzmodule bilden den zentralen Bestandteil von Verhaltensnetzwerken und ähneln klassischen Strips-Operatoren [Fikes und Nilsson, 1971]. Sie sind durch ein Tupel (C, A, D, α) definiert. Dabei ist

- C$\subseteq \mathcal{P}$ eine Menge von *Vorbedingungen* c, die erfüllt sein müssen, damit das Modul ausgeführt werden kann,
- A $\subseteq \mathcal{P}$ eine Menge von *erzeugenden*, d.h. eine Proposition wahr machenden *Effekten* a, die nach der Ausführung des Verhaltens erwartet werden,

- D $\subseteq \mathcal{P}$ eine Menge von *vernichtenden*, d.h. eine Proposition falsch machenden *Effekten* d und

- α die *Aktivierung* des Moduls,

wobei $\mathcal{P}$ die Menge aller Situations-Propositionen bildet.

Sei $S^t \subseteq \mathcal{P}$ die Teilmenge der Propositionen, die zum Zeitpunkt t wahr sind, $G^t \subseteq \mathcal{P}$ die Propositionen, die zum Zeitpunkt t Zielzustände des Agenten darstellen (Erreichungsziele) und $R^t \subseteq \mathcal{P}$ Zielzustände, die zum Zeitpunkt t bereits erfüllt sind (geschützte Ziele oder Erhaltungsziele). Die Mengen S, G und R sind dabei durch die Umgebung festgelegt, sind also aus Sicht des Agenten nicht beeinflußbar.

Die Menge von Parametern zur Steuerung der Aktivierungsausbreitung (Abschnitt 2.2) und der Verhaltensauswahl (Abschnitt 2.3) ist dann gegeben durch

- π, der mittleren Gesamtaktivierung im Netzwerk,

- θ, der *Aktivierungsschwelle*, die die Aktivierung α eines Kompetenzmoduls überschreiten muß, um ausgeführt zu werden,

- ϕ, die Stärke der Aktivierung, die ein Kompetenzmodul von Propositionen in S^t erhält,

- γ, die Stärke der Aktivierung von Propositionen in G^t (den Zielen),

- δ, die Stärke der Hemmung von Propositionen in R^t (Erhaltungsziele).

2.2 Aktivierungsmechanismus

Die Vorbedingungen verschiedener Kompetenzmodule in einem Verhaltensnetzwerk müssen nicht disjunkt sein, d.h. es können mehrere Kompetenzmodule gleichzeitig ausführbar sein. Für die Verhaltensauswahl benötigt man daher ein Kriterium, bzw. eine Ordnung, die angibt, welches der ausführbaren Kompetenzmodule ausgeführt werden soll. Diese Ordnung ist durch die Aktivierung der Kompetenzmodule gegeben. Die Aktivierung entscheidet zudem, ob überhaupt ein Verhalten ausgeführt wird, da nur Kompetenzmodule ausgeführt werden, deren Aktivierung größer als die Aktivierungsschwelle θ ist.

Zur Berechnung der Aktivierung sind die Kompetenzmodule in einem Netzwerk miteinander und mit den Ziel- und Situationspropositionen verbunden [Maes, 1989; 1990a]. Diese Verbindungen können sowohl aktivierend als auch hemmend auf die Aktivierung eines Kompetenzmodule einwirken.

Sei $M_c \subseteq \mathcal{K}$ mit $c \in C_k$ die Menge von Kompetenzmodulen, die c als Vorbedingung enthalten, $N_a \subseteq \mathcal{K}$ mit $a \in A_k$ die Menge von Kompetenzmodulen, die a als erzeugenden Effekt enthalten, $U_d \subseteq \mathcal{K}$ mit $d \in D_k$ die Menge von Kompetenzmodulen, die d als vernichtenden Effekt enthalten und $E \subseteq \mathcal{K}$ die Menge der ausführbaren Kompetenzmodule, d.h. der Module, für die $C \subseteq S^t$ ist.

Ein Kompetenzmodul k erhält dann für jede erfüllte Vorbedingung die Aktivierung

$$\alpha_{k,s}^{t} = \phi \sum_{c \in C \cap \mathrm{S}^t} \frac{1}{\mid \mathrm{M}_c \mid \cdot \mid \mathrm{C} \mid} \tag{2.1}$$

Um Module mit wenigen Vorbedingungen nicht zu benachteiligen wird die Aktivierung durch die Anzahl der Vorbedingungen | C | geteilt (Inputnormierung). Außerdem wird die Aktivierung, die eine Situationsproposition ins Netzwerk einspeist, unter allen | M_c | empfangenden Kompetenzmodulen aufgeteilt (Fan-Effekt). Die Aktivierung durch Situationspropositionen dient dazu, Kompetenzmodule mit einem hohen Anteil wahrer Vorbedingungen insofern zu fördern, als diese mehr Aktivierung erhalten, die sie an Vorgänger, also an Module, die diese nicht erfüllten Vorbedingungen wahr machen, weitergeben können.

Ein Kompetenzmodul erhält für jeden erzeugenden Effekt Aktivierung von Zielen, d.h. von Propositionen in G^t

$$\alpha_{k,g}^{t} = \gamma \sum_{a \in A \cap \mathrm{G}^t} \frac{1}{\mid \mathrm{N}_a \mid \cdot \mid \mathrm{A} \mid} \tag{2.2}$$

Wieder wird die Aktivierung einer Zielproposition unter allen | N_a | empfangenden Kompetenzmodulen aufgeteilt sowie durch die Anzahl | A | der Effekte des Moduls geteilt. Dies führt jedoch zu unerwünschten Effekten, wie in Kapitel 2.5.2 näher erläutert wird.

Durch jeden Effekt d, der ein Erhaltungsziel vernichtet, reduziert sich die Aktivierung um

$$\alpha_{k,p}^{t} = \delta \sum_{d \in D \cap \mathrm{R}^t} \frac{1}{\mid \mathrm{U}_d \mid \cdot \mid \mathrm{D} \mid} \tag{2.3}$$

Neben der Aktivierung von außen tauschen Kompetenzmodule auch Aktivierung untereinander aus. Ein Kompetenzmodul k erhält für jede nicht erfüllte Vorbedingung c von jedem diese Bedingung erfüllenden und ausführbaren Modul l (Vorgängermodul) die Aktivierung

$$\alpha_{k,c}^{t} = \frac{\phi}{\gamma} \sum_{l \in \mathrm{E} \setminus \{k\}} \sum_{c \in (\mathrm{C}_k \cap \mathrm{A}_l) \setminus \mathrm{S}^t} \frac{1}{\mid \mathrm{M}_c \mid \cdot \mid \mathrm{C} \mid} \tag{2.4}$$

Es erhält für jeden erfüllenden Effekt a von jedem Modul l, das a als zum Zeitpunkt t nicht erfüllte Vorbedingung enthält (Nachfolgermodul), die Aktivierung

$$\alpha_{k,a}^{t} = \sum_{l \in \mathrm{K} \setminus \mathrm{E} \setminus \{k\}} \sum_{a \in (\mathrm{A}_k \cap \mathrm{C}_l) \setminus \mathrm{S}^t} \frac{\alpha_k^{t-1}}{\mid \mathrm{N}_a \mid \cdot \mid \mathrm{A} \mid} \tag{2.5}$$

Die Aktivierung wird durch jeden Effekt d, von jedem Modul l, das d als Vorbedingung enthält (Konfliktormodul), vermindert um

$$\alpha_{k,d}^{t} = \frac{\delta}{\gamma} \sum_{l \in \mathrm{K} \setminus \{k\}} \sum_{d \in (\mathrm{D}_k \cap \mathrm{C}_l) \cap \mathrm{S}^t} \frac{\alpha_k^{t-1}}{\mid \mathrm{U}_d \mid \cdot \mid \mathrm{D} \mid}, \tag{2.6}$$

falls die Aktivierung $\alpha_l^{t-1} > \alpha_k^{t-1}$ ist. Gegenseitige Hemmung wird dadurch ausgeschlossen. Dies soll verhindern, daß sich die aktivsten Kompetenzmodule gegenseitig eliminieren.

Die interne Aktivierung und Hemmung der Kompetenzmodule untereinander hat dieselbe Semantik wie Aktivierung und Hemmung von außen. Effekte von ausführbaren Kompetenzmodulen entsprechen Vorhersagen für zukünftige Zustände, nicht erfüllte Vorbedingungen entsprechen Unterzielen des Netzwerks, und bereits erfüllte Vorbedingungen entsprechen Erhaltungszielen. Die Faktoren der internen Aktivierung sind so gewählt, daß die Verhältnisse von Vorgänger-, Nachfolger- und Konfliktor-Aktivierung denen von Situations-, Ziel- und Erhaltungsziel-Aktivierung entsprechen.

Mit Hilfe der Parameter ist es möglich, das Verhalten des Agenten eher zielorientiert oder eher situationsorientiert zu gestalten. Ein hohes γ erhöht z.B. den Einfluß der Zielpropositionen auf die Aktivierung, wodurch zielgerichtete Verhalten stärker bevorzugt werden. Gleichzeitig legt γ das Verhältnis von interner zu externer Aktivierung fest. Eine hohes ϕ steigert dagegen den Einfluß der Situation.

Die Gesamtaktivierung eines Kompetenzmoduls k zum Zeitpunkt t ist

$$\alpha_{k,\Sigma}^t = \alpha_{k,s}^t + \alpha_{k,g}^t - \alpha_{k,p}^t + \alpha_{k,c}^t + \alpha_{k,a}^t - \alpha_{k,d}^t. \tag{2.7}$$

Diese Gesamtaktivierung wird dann noch so normiert, daß die mittlere Aktivierung der Kompetenzmodule konstant π beträgt:

$$\alpha_k^t = \pi \frac{\mid \mathrm{K} \mid \alpha_{k,\Sigma}^t}{\sum_{l \in K} \alpha_l^t} \tag{2.8}$$

Da für diese Berechnung die Aktivierung aller anderen Kompetenzmodule bekannt sein muß, ist hier das Lokalitätsprinzip der Berechnung verletzt. D.h. nicht alle Information die ein Kompetenzmodul zur Berechnung der Aktivierung benötigt ist lokal im Modul gespeichert oder kann über Aktivierungsverbindungen ermittelt werden. Die vollständige Information ist nur verfügbar, wenn jedes Modul mit jedem anderen verbunden ist, was jedoch nicht der Fall ist.

2.3 Verhaltensauswahl

Die eigentliche Verhaltensauswahl erfolgt in vier Schritten [Maes, 1990b]:

1. Der Einfluß von Erreichungs- und Erhaltungszielen sowie von der momentanen Situation auf die Kompetenzmodule wird berechnet (Gl. 2.1 - 2.3).

2. Die gegenseitig Aktivierung und Hemmung der Kompetenzmodule wird bestimmt (Gl. 2.4 - 2.6).

3. Die Aktivierung aller Kompetenzmodule wird so verringert, daß die Gesamtaktivierung im Netzwerk konstant bleibt (Gl. 2.8).

4. Ist die Aktivierung des ausführbaren Kompetenzmoduls mit der höchsten Aktivierung aller ausführbaren Module größer als die Aktivierungsschwelle θ wird es ausgeführt, die Aktivierung auf Null und θ auf seinen ursprünglichen Wert zurückgesetzt. Erfüllt keines der Kompetenzmodule diese Bedingungen, wird θ um 10% reduziert und wieder bei 1. begonnen.

Der Parameter θ sorgt dafür, daß ausreichend lange Aktivierungsaustausch stattfindet. Durch die Weitergabe von Aktivierung innerhalb der Kompetenzmodule werden je Selektionszyklus um ein Verhalten längere Verhaltensfolgen berücksichtigt. Je höher also θ ist, desto längere Verhaltensfolgen werden berücksichtigt und desto vorausschauender ist das Verhalten des Agenten. Je niedriger θ ist, desto schneller reagiert der Agent.

2.4 Parametereinstellung

Die globalen Parameter Π eines Verhaltensnetzwerks sind domänenabhängig, müssen also für jede Domäne angepaßt werden. In [Maes, 1991] wird beschrieben, wie diese Parameter mit Hilfe eines Meta-Netzwerks automatisch eingestellt werden können. Dazu wird zusätzlich zum Verhaltensnetzwerk für die Handlungskontrolle ein sogenanntes Meta-Netzwerk eingeführt, dessen Kompetenzmodule nicht Einfluß auf das Verhalten des Agenten, sondern auf die Parameter des Verhaltensnetzwerks haben. So kann z.B. eine Verhaltensregel darin bestehen, den Agenten schneller entscheiden zu lassen, wenn eine sehr dynamische Umgebung vorliegt, indem die Aktivierungsschwelle θ vermindert wird. Insgesamt verwendet Maes neun Kompetenzmodule für das Meta-Netzwerk.

Die initiale Parametereinstellung für das Verhaltensnetzwerk erfolgt zufällig, die Parametereinstellung für das Meta-Netzwerk selbst wird manuell vorgenommen. Das läßt sich zum einen rechtfertigen durch die Einfachheit des Meta-Netzwerks, das vollständig situationsgesteuert ist, also keine Ziele besitzt. Zum anderen ist die Domäne des Meta-Netzwerks immer dieselbe, nämlich die Kontrolle der Parameter eines anderen Verhaltensnetzwerks. Damit sind die Parameter des Meta-Netzwerks unabhängig von der Domäne des Agenten [Maes, 1991].

Für die empirischen Untersuchungen in der RoboCup-Umgebung war es jedoch wichtig, alle Parameter der Netzwerke zu kontrollieren, um identische Versuchsbedingungen garantieren zu können. Dies wäre durch ein automatisches Adaptieren der Netzwerkparameter nicht zu gewährleisten und würde zu größeren zufälligen Streuungen der Versuchsergebnisse führen. Daher wurden die Parameterwerte, wie in Abschnitt 5.1 beschrieben, durch Vorversuche bestimmt und in den eigentlichen Versuchen konstant gehalten.

Da im Rahmen dieser Arbeit nur zwei Domänen untersucht wurden, war der Arbeits- und Zeitaufwand für diese Vorversuche noch vertretbar. Die Unabhängigkeit von Verhaltensnetzwerken von speziellen Domänen ist aber zweifelsohne nur durch automatische Anpassung der Parameter zu gewährleisten.

2.5 Probleme und Einschränkungen

Neben den vielen Vorzügen von derartigen Verhaltensnetzwerken wie der Berücksichtigung mehrerer gleichzeitiger Zielpropositionen, Reaktivität, Stabilität, parallele Berechnung, gibt es auch einige Probleme und Einschränkungen sowohl bei der Aktivierungsausbreitung als auch bei der Verhaltensauswahl.

2.5.1 Aktivierungsrückkopplung

Zwischen allen Kompetenzmodulen, die über eine Vorgängerverbindung verknüpft sind, gibt es auch eine Nachfolgerverbindung über die Aktivierung ausgetauscht werden kann. Ist ein Nachfolgermodul nicht ausführbar, wird über beide Verbindungen Aktivierung übertragen. Tyrrell [1994] wies darauf hin, daß durch eine solche Rückkopplung Vorgängermodule (appetitive Verhalten) gegenüber direkt zielerfüllenden Kompetenzmodulen (konsummatorische Verhalten) bevorzugt werden (siehe Abb. 2.1). In nicht-deterministischen Umgebungen wäre erwünscht, daß immer Module bevorzugt werden, die ein Ziel direkt erfüllen. Wie in Kapitel 3.2 dargestellt wird, kann die Aktivierung eines Kompetenzmoduls in erweiterten Verhaltensnetzwerken auch als dessen erwarteten Nutzen verstanden werden. Der erwartete Nutzen (eines Verhaltens) ergibt sich als Summe aller Produkte von Erwartungswahrscheinlichkeit des Eintreffens eines Effekts und dessen Wert $eu = \sum_i p_i \cdot u_i$ ergibt . Der erwartete Nutzen $eu_3 = p \cdot u(goal)$ von 'action3' in Abbildung 2.1 sollte gleich oder größer sein als der erwartete Nutzen $eu_2 = p^2 \cdot u(goal)$ von 'action2'. p ist dabei die Wahrscheinlichkeit, mit der ein Effekt eintritt und ist in MASM-Netzwerken eins. Die Aktivierung in MASM-Netzwerken entspricht hier nicht dem Erwartungs-Wert Prinzip der Entscheidungstheorie, da die Aktivierung von 'action3' kleiner ist als die Aktivierung von 'action2'[1].

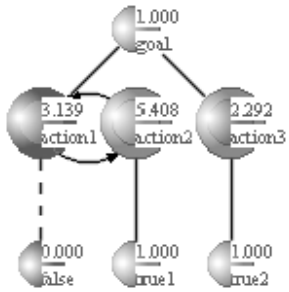

Abbildung 2.1: Bevorzugung des appetitiven Verhaltens 'action2' gegenüber dem konsummatorischen Verhalten 'action3' durch Aktivierungsrückkopplung. Oben dargestellt ist das Ziel, in der Mitte die Kompetenzmodule und deren Aktivierung und unten die Situationspropositionen.

[1] Formal ist die Aktivierung von 'action2' gegeben durch $\alpha_2^t = r(\alpha_2^{t-1}) + \gamma + \phi + \phi/\gamma\alpha_2^{t-2} + \ldots$, die von 'action3' durch $\alpha_3^t = r(\alpha_3^{t-1}) + \gamma + \phi$. Da $\gamma, \phi > 0$ und r() monoton wachsend ist, ist $\alpha_2^t > \alpha_3^t$ unabhängig von γ, ϕ.

Neben ausschließlich aktivierender Rückkopplung über Vorgänger-Nachfolger und Nachfolger-Nachfolger Verbindungen kann es auch Rückkopplungsschleifen geben, die sowohl aktivierende als auch hemmende Verbindungen enthalten. Diese Art der Rückkopplung kann in einigen abgewandelten Versionen von Verhaltensnetzwerken (Variation vier in [Tyrrell, 1994]) zu oszillierender Aktivierung führen.

Ebenso kann auch ausschließlich hemmende Rückkopplung zwischen drei oder mehr Kompetenzmodulen eintreten. Die gegenseitige Hemmung zweier Kompetenzmodule wird allerdings ausgeschlossen (Gleichung 2.6).

2.5.2 Aktivierungsverteilung

Die Kompetenzmodule teilen in MASM-Netzwerken die eingehende Aktivierung durch die Anzahl Vorbedingungen bzw. erreichender bzw. vernichtender Effekte (Inputnormierung, Gleichungen 2.1 - 2.6). Dadurch soll verhindert werden, daß Module mit vielen Vorbedingungen bzw. Effekten mehr Aktivierung erhalten als Module mit wenigen [Maes, 1990b]. Es ist aber fraglich, warum Kompetenzmodule mit vielen erwünschten Effekten nicht mehr Aktivierung erhalten sollen, als solche mit wenigen. Tatsächlich werden Kompetenzmodule, die mehrere Ziele gleichzeitig erreichen, nicht bevorzugt (Abb. 2.2). Module, die mehrere Erhaltungsziele zerstören, werden nicht benachteiligt. Damit verstoßen MASM-Netzwerke auch hier gegen das Erwartungs-Wert Prinzip der Entscheidungstheorie. Der erwartete Nutzen von 'action1' in Abbildung 2.2 ist entsprechend $eu_1 = p \cdot u(goal1)$, der erwartete Nutzen von 'action2' beträgt $eu_2 = p \cdot u(goal2) + p \cdot u(goal3)$ und ist somit größer als eu_1, da hier $u(goal1) = u(goal2) = u(goal3)$ und $p = 1$ ist.

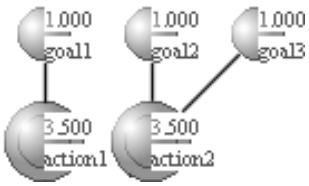

Abbildung 2.2: Kompetenzmodule, die mehrere Ziele gleichzeitig erreichen, werden aufgrund der Inputnormierung nicht bevorzugt.

Ebenso wird in MASM-Netzwerken die von Situations-, Zielpropositionen und Kompetenzmodulen ausgehende Aktivierung durch die Anzahl der empfangenden Kompetenzmodule geteilt (Fan-Effekt). Damit soll erreicht werden, daß Kompetenzmodule, die dasselbe Ziel erreichen, bzw. eine gemeinsame Situationsproposition in ihrer Vorbedingung haben, im gegenseitigen Wettstreit liegen, ausgeführt zu werden [Maes, 1990b]. Es führt aber auch dazu, daß Ziele, die durch mehrere Verhalten erreicht werden können, gegenüber Zielen, die nur von einem Verhalten erreicht werden, benachteiligt werden, auch wenn beide Ziele gleich wichtig sind (Abb. 2.3) [Tyrrell, 1994]. Auch dies ist ein Verstoß gegen das Erwartungs-Wert Prinzip der Entscheidungstheorie. Der erwartete

Nutzen von 'action1' $eu_1 = p \cdot u(goal1)$ in Abbildung 2.3 sollte mit dem erwarteten Nutzen von 'action2' und 'action3' $eu_{2/3} = p \cdot u(goal2)$ identisch sein (da $u(goal1) = u(goal2)$).

Abbildung 2.3: Ziele, die von mehreren Verhaltensmodulen erfüllt werden, werden von MASM durch den Fan-Effekt benachteiligt.

Tyrrell [1994] zeigt jedoch, daß durch ein einfaches Weglassen des Fan-Effekts neue Probleme entstehen. Unter anderem können Kompetenzmodule mit vielen Nachfolgerverbindungen viel Aktivierung von Kompetenzmodulen erhalten, die nur Alternativen desselben Ziels darstellen (Abb. 2.4). Es läßt sich also prinzipiell nicht mehr entscheiden, ob die Aktivierung von einem oder von mehreren Ziel- bzw. Situationspropositionen stammt. Die Aktivierung ist sowohl von der Anzahl der Ziele im Netzwerk als auch von der Anzahl Kompetenzmodule abhängig.

Entscheidungstheoretisch stellt sich das Problem wie folgt dar: der maximale erreichbare Nutzen von 'action3' im linken Beispiel beträgt $e\hat{u}_{3,l} = u(goal1)$, im rechten Beispiel beträgt er $e\hat{u}_{3,r} = u(goal1) + u(goal2)$. Die Aktivierung von 'action3' rechts sollte also höher sein, als die Aktivierung von 'action3' links. Beim einfachen Weglassen des Fan-Effekts erhalten aber beide Kompetenzmodule dieselbe Aktivierung. Der Agent ist indifferent zwischen den beiden Verhalten.

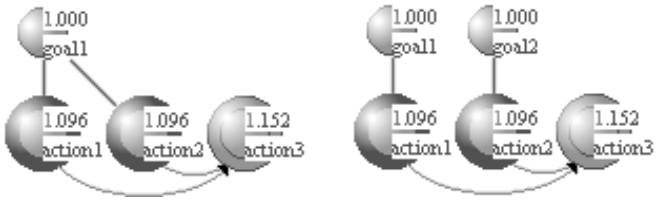

Abbildung 2.4: Ein appetitives Verhalten (action3) kann ohne Fan-Effekt nicht unterscheiden, ob die erhaltene Aktivierung ursprünglich von einem Ziel (links) oder von mehreren Zielen (rechts) stammt.

2.5.3 Parallele Verarbeitung

Für die parallele Informationsverarbeitung der Kompetenzmodule ist es notwendig, das Lokalitätsprinzip einzuhalten. Das Lokalitätsprinzip besagt, daß für die Informationsverarbeitung in einem Knoten nur Information verwendet werden darf, die im Knoten lokal gespeichert ist, oder die der Knoten über Verbindungen zu anderen Knoten erhalten kann.

Maes führt an [Maes, 1990a; 1990b; 1992], daß der Prozeß der Handlungskontrolle lokal in den Kompetenzmodulen stattfindet und damit grundsätzlich parallelisierbar ist. In folgenden drei Punkten trifft dies aber nicht zu:

1. Für die Division der eingehenden Aktivierung durch die Anzahl empfangender Kompetenzmodule (Gl. 2.1-2.6) ist es notwendig, diese Anzahl zu kennen. Nachdem ein Kompetenzmodul bei der Erstellung des Netzwerks diese Anzahl kennt, kann zwar die Aktivierung lokal berechnet werden. Das Hinzufügen neuer Kompetenzmodule hat dann aber Änderungen in jedem Kompetenzmodul zur Folge. Jedes Kompetenzmodul hängt von der gesamten Struktur des Netzwerks ab, also auch von Knoten, zu denen es keine Verbindung hat.

2. Für die Normierung der Aktivierung eines Kompetenzmoduls (Gl. 2.8) muß die Gesamtanzahl Knoten des Netzwerks bekannt sein, was zu den selben Problemen wie bei 1. führt. Problematischer ist aber, daß auch die sich ständig ändernde Gesamtaktivierung des Netzwerks für die Normierung benötigt wird. Dadurch ist die Berechnung der Aktivierung eines Kompetenzmoduls nicht mehr lokal durchführbar.

3. Auch die Verhaltensauswahl (Kapitel 2.3) verstößt gegen das Lokalitätsprinzip. Für die Auswahl des aktivsten Verhaltens muß die Aktivierung aller Kompetenzmodule bekannt sein. Diese Information ist aber global, d.h. ein Kompetenzmodul kann nicht lokal entscheiden, ob es ausgeführt werden soll oder nicht.

Streng genommen verstoßen auch die globalen Netzwerkparameter Π gegen das Lokalitätsprinzip, da jedes Kompetenzmodul Zugriff auf die Parameter haben muß. Die einfachste Lösung, die Speicherung aller Parameter in jedem Knoten, ist unbefriedigend, da insbesondere die Aktivierungsschwelle θ ständigen Änderungen unterworfen ist, die nicht lokal berechnet werden können. So wird θ bei der Ausführung eines Verhaltens auf den ursprünglichen Wert zurückgesetzt. Die Information, daß ein Verhalten ausgeführt wurde, ist den Kompetenzmodulen aber nicht lokal verfügbar. Das Problem der globalen Netzwerkparameter kann aber dennoch relativ einfach gelöst werden, indem man für jeden Parameter einen Netzwerkknoten einführt, der mit jedem Verhalten verbunden ist.

In Kapitel 3 wird beschrieben, wie die gesamte Aktivierungsausbreitung und die Verhaltensauswahl lokal durchgeführt werden kann, ohne daß eine vollständige Verknüpfung aller Kompetenzmodule notwendig ist.

2.5.4 Gleichzeitige Ausführung mehrerer Verhalten

Während für die Auswahl eines Verhaltens mehrere Ziele gleichzeitig berücksichtigt werden, erlaubt der Algorithmus zur Verhaltensauswahl (Kapitel 2.3) nur die Ausführung eines Verhaltens zu einem Zeitpunkt. Menschen sind jedoch in der Lage, gut trainierte Verhalten gleichzeitig auszuführen, sofern diese nicht dieselben Ressourcen beanspruchen [Navon und Gopher, 1979]. Die Kompetenzmodule enthalten jedoch keine Information über die beanspruchten Ressourcen

des zugehörigen Verhaltens. Die gleichzeitige Ausführung von Verhalten, die dieselbe Ressource verwenden, führt in der Regel dazu, daß keines der Verhalten erfolgreich durchgeführt werden kann. Beispielsweise kann bei Menschen Unentschlossenheit zwischen dem Sprechen von zwei Wörtern 'close' und 'shut' zum Sprechen eines nicht existierenden Worts 'clut' führen [Norman, 1981]. Die gemeinsame Ressource 'Sprachapparat' kann nicht gleichzeitig von mehreren Verhalten verwendet, aber gleichwohl von verschiedenen Intentionen beeinflußt werden.

Wie in Kapitel 2.3 beschrieben, kommt in MASM-Netzwerken maximal ein Verhalten zu einem Zeitpunkt zur Ausführung. Die sequentielle Ausführung von Verhalten in MASM-Netzwerken vermeidet das Problem von Ressourcenkonflikten. Der Preis ist zum einen eine geringere Performanz bei Aufgaben, die die parallele Ausführung von Verhalten erlauben. Zum anderen können Aufgaben, die die gleichzeitige Ausführung von Verhalten notwendig erfordern gar nicht ausgeführt werden.

Kapitel 3

Erweiterte Verhaltensnetzwerke

Wie im vorigen Kapitel beschrieben entsprechen MASM-Verhaltensnetzwerke bereits in einigen sehr wichtigen Punkten den in Kapitel 1.1.3 genannten Forderungen an Handlungskontrolle.

- Sie sind aufgrund des Mechanismus der Aktivierung und Hemmung von Verhalten durch Ziele in der Lage, mehrere, auch in Konflikt zueinander stehende Ziele gleichzeitig bei der Verhaltensauswahl zu berücksichtigen.
- Sie erlauben es, aufgrund der geringen Berechnungskomplexität schnell und aufgrund der situationsabhängigen Verhaltensauswahl der aktuellen Situation angemessen zu agieren. Die Reaktivität und Zielgerichtetheit des Verhaltens kann darüber hinaus mittels Parameter gesteuert werden.
- Die situationsabhängige Verhaltensauswahl macht MASM-Netzwerke robust gegenüber nicht-deterministischen Effekten von Verhalten.
- Aktivierungsverbindungen in MASM-Netzwerken können in diskreten Umgebungen gelernt bzw. adaptiert werden.

Damit sind sie in dynamischen und nicht-deterministischen Umgebungen deliberativen Ansätzen zur Handlungskontrolle überlegen, da sie schnell und situationsangepaßt reagieren können. Sie sind reaktiven Ansätzen überlegen, da sie Ziele explizit repräsentieren können und zielgerichtetes Verhalten bevorzugen.

Dennoch erfüllen MASM-Netzwerke nur einen Teil der Forderungen an Handlungskontrolle in komplexen Umgebungen. Im folgenden wird erläutert inwieweit die Forderungen aus Kapitel 1.1.3 von MASM-Netzwerken nicht erfüllt werden. Gleichzeitig wird auf die entsprechenden Verbesserungen und Erweiterungen verwiesen, die in diesem Kapitel vorgestellt werden.

Forderung1: zielgerichtet handeln

MASM-Netzwerke sind zwar in der Lage, mehrere Ziele bei der Verhaltensauswahl zu berücksichtigen. Allerdings werden Verhalten, die mehreren Zielen

gleichzeitig zuträglich sind, nicht unbedingt bevorzugt (Kapitel 2.5.2). Aus entscheidungstheoretischen Gesichtspunkten sollte eine Option, die einen höheren erwarteten Nutzen hat, eine höhere Bewertung erhalten und entsprechend bevorzugt werden (siehe Kapitel 1.2). Der Aktivierungsmechanismus von erweiterten Verhaltensnetzwerken wurde so geändert, daß die Aktivierung durch Ziele den entscheidungstheoretischen erwarteten Nutzen angibt. Dazu wurden zum ersten als einzige Quelle für Aktivierung die Ziele zugelassen. Aktivierung durch die Situation, wie sie bei MASM-Netzwerken stattfindet, ist in erweiterten Verhaltensnetzwerken nicht vorhanden (siehe Kapitel 3.2.5). Zum zweiten wurde ein neues Verfahren zur Aktivierungsweitergabe eingeführt, bei dem unter Vermeidung von Aktivierungsrückkopplung (Kapitel 2.5.1) und Aktivierungszusammenfluß (Kapitel 2.5.2) die Aktivierung eines Kompetenzmoduls den maximal erreichbaren Nutzen des Verhaltens berücksichtigt. (siehe Kapitel 3.2.3). Zum dritten können Ziele bei erweiterten Verhaltensnetzwerken nach ihrer Wichtigkeit priorisiert werden. Die Wichtigkeit eines Ziels bestimmt dessen maximalen Nutzen.

Trotz der unterschiedlichen Wichtigkeit von Motiven oder Zielen dominieren nicht immer die wichtigsten das Verhalten. In biologischen Systemen wird z.B. der subjektive Wert des Motivs satt zu sein höher eingeschätzt und beeinflußt dementsprechend stärker das Verhalten, je mehr Zeit seit der letzten Nahrungsaufnahme vergangen ist [Becker–Carus, 1983]. Der Nutzen eines Ziels ist also nicht unbedingt statisch, sondern kann von der Situation abhängen. Mit der Einführung von Relevanzbedingungen für Ziele in erweiterten Verhaltensnetzwerken wurde die Möglichkeit eines situationsabhängigen Einflusses der Ziele auf die Verhaltenskontrolle geschaffen.

Forderung2: situationsangepaßt handeln und schnell reagieren

Um der Situation angemessen zu handeln ist es zum einen nötig, die Situation möglichst genau zu kennen. Zum anderen muß die Kenntnis der aktuellen Situation ständigen Einfluß auf die Verhaltensauswahl haben. Letzteres wird in Verhaltensnetzwerken durch direkte Verbindung von Wahrnehmungsknoten und Kompetenzmodulen und durch einen kurzen Entscheidungszyklus erreicht. Um auch in kontinuierlichen Umgebungen die Situation möglichst genau zu erfassen ohne die Zahl der nötigen Propositionen unnötig zu erhöhen, wurden in erweiterten Verhaltensnetzwerken kontinuierliche Wahrheitswerte der Propositionen eingeführt (siehe Kapitel 3.3.3). Kontinuierliche Zustände können so genauer repräsentiert werden als es mit binärwertigen Propositionen der Fall ist. Dadurch lassen sich auch kontinuierliche Verhaltensregeln der Art formulieren: 'Je relevanter das Ziel ist, desto intensiver führe das Verhalten aus' (siehe Kapitel 3.3.4).

Für ein schnelles Handeln, bzw. für eine schnelle Reaktion auf unvorhergesehene Ereignisse ist es notwendig, daß der Entscheidungsprozeß wenig Zeit in Anspruch nimmt. Durch eine parallele Berechnung der Verhaltensauswahl kann die Berechnungskomplexität gesenkt und dadurch der Entscheidungsprozeß beschleunigt werden. Tatsächlich beruht die Leistungsfähigkeit des menschlichen Gehirns auf der massiv parallelen Informationsverarbeitung der Neuronen. An-

statt vieler hintereinandergeschalteter Neuronen verrichten Millionen von Nervenzellen parallel ihre Arbeit. Bei motorischen Reflexen sind z.B. zumeist nur drei Neuronen hintereinander geschaltet [Strube, 1998]. Verschiedene Reflexe werden aber parallel und unabhängig voneinander verarbeitet. Damit soll nicht der Anspruch erhoben werden, die Informationsverarbeitung in erweiterten Verhaltensnetzwerken sei ein Modell für neuronale Informationsverarbeitung beim Menschen. Die Informationsverarbeitung in Kompetenzmodulen ist tatsächlich weitaus komplexer als die einer Nervenzelle. Es soll lediglich das gemeinsame Prinzip der parallelen Informationsverarbeitung zur Verringerung der Berechnungskomplexität betont werden.

Wie in Kapitel 2.5.3 beschrieben verhindern Verstöße gegen das Lokalitätsprinzip, daß der Entscheidungsprozeß in MASM-Netzwerken vollständig parallel durchgeführt werden kann. Mit der Einführung des Goaltracking in erweiterten Verhaltensnetzwerken (siehe Kapitel 3.2.3), das die Semantik der Aktivierungsweitergabe von MASM-Netzwerken ersetzt, wurde einer der Verstöße gegen das Lokalitätsprinzip beseitigt. Mit dem Mechanismus zur parallelen Verhaltensausführung in erweiterten Verhaltensnetzwerken (siehe Kapitel 3.3) wurde gleichzeitig ein weiterer Verstoß von MASM-Netzwerken beseitigt, so daß der komplette Entscheidungsprozeß in erweiterten Verhaltensnetzwerken parallel in den Knoten des Netzwerks durchgeführt werden kann.

Forderung3: parallele Verhaltensausführung

Wie in Kapitel 2.5.4 ausgeführt, sind MASM-Verhaltensnetzwerke auf die Ausführung eines Verhaltens zu einem Zeitpunkt beschränkt. Sollen Verhalten zeitgleich ausgeführt werden, benötigt der Agent Wissen über die von den jeweiligen Verhalten benötigten Ressourcen. Beanspruchen Aufgaben unterschiedliche Ressourcen, können sie (ohne Performanzeinbuße) parallel ausgeführt werden [Shaffer, 1975; Navon und Gopher, 1979]. Durch die gleichzeitige Ausführung von Verhalten kann ein Ziel schneller, bzw. können mehrere Ziele gleichzeitig erreicht werden.

Um in erweiterten Verhaltensnetzwerken Verhalten parallel auszuführen, benötigen die Kompetenzmodule also Information über die von ihnen verwendeten Ressourcen. Dazu wird zunächst in Kapitel 3.1 der Begriff der Ressource eingeführt und Kompetenzmodule um die entsprechende Information erweitert. Für die zeitgleiche Ausführung von Verhalten muß dann der Zugriff auf Ressourcen koordiniert werden. In Kapitel 3.3 wird beschrieben, wie unter Verwendung von Ressourceknoten in erweiterten Verhaltensnetzwerken parallele Verhaltensausführung erreicht wird.

Im folgenden wird zunächst der Aufbau und Formalismus erweiterter Verhaltensnetzwerke (EBN **e**xtended **b**ehavior **n**etworks) vorgestellt. Dann wird der in weiten Teilen geänderte Aktivierungsmechanismus vorgestellt und erläutert. Danach wird der Algorithmus zur parallelen Verhaltensauswahl beschrieben und diskutiert. Das Kapitel endet mit einem Beispiel zur Erläuterung der Aktivierungsmechanismen eines erweiterten Verhaltensnetzwerks.

3.1 Aufbau

Sei $\mathcal{S}$ eine Menge von Weltzuständen, $\mathcal{P}^+$ eine Menge von Atomen und $\tau_P : \mathcal{P}^+ \times \mathcal{S} \rightarrow [0..1]$ eine Funktion, die jedem Atom aus $\mathcal{P}^+$ in jedem Zustand einen (kontinuierlichen) Wahrheitswert zuweist. $\mathcal{P}$ sei eine Menge von Atomen und negierten Atomen, wobei $\tau_P(\neg p, s) := 1 - \tau_P(p, s)$. $\mathcal{L}^\wedge$ ist eine propositionale Sprache über $\mathcal{P}$ und der logischen Verknüpfung $\wedge$, wobei $\tau_P(p \wedge q, s) := \tau_P(p, s) \otimes \tau_P(q, s)$ und $\otimes$ eine beliebige kontinuierliche T-Norm ist (z.B. $min(p, q)$, $p \cdot q$). $\mathcal{L}$ ist eine propositionale Sprache über $\mathcal{P}$ und den logischen Verknüpfungen $\wedge$ und $\vee$, wobei $\tau_P(p \vee q, s) := \tau_P(p, s) \oplus \tau_P(q, s)$ und $\oplus$ eine beliebige kontinuierliche T-Konorm (z.B. $max(p, q)$, $x + y - xy$) [Saffiotti *et al.*, 1995].

Definition 1 *Ein* Ziel *besteht aus einem Tupel (*GCon, ι, RCon*) mit*

- GCon $\in \mathcal{L}^\wedge$ *der* Zielbedingung, *d.h. den Situationen, in denen das Ziel erfüllt ist,*
- $\iota \in [0..1]$ *der (statischen)* Wichtigkeit *des Ziels,*
- RCon $\in \mathcal{L}$ *der* Relevanzbedingung, *d.h. der situationsabhängigen (dynamischen) Wichtigkeit des Ziels mit* $r = \tau_P(RCon, s)$ *der* Relevanz *des Ziels.*

Sei weiterhin $\mathcal{R}$ die Menge aller Ressourcen und $\tau_R : \mathcal{R} \times \mathcal{S} \rightarrow \mathbb{R}^+$ eine Funktion, die jedem Element aus $\mathcal{R}$ die Menge im Zustand s vorhandener Ressourceeinheiten zuweist. Die Funktion $\tau_U : \mathcal{M} \times \mathcal{R} \times \mathcal{S} \rightarrow \mathbb{R}^+$, mit $\mathcal{M}$ der Menge aller Kompetenzmodule (siehe unten), gibt die erwartete Menge an benötigten Ressourceeinheiten eines Kompetenzmoduls zur Erreichung der zugehörigen Effekte in einer Situation an.

Definition 2 *Ein* Ressourceknoten *ist ein Tupel (res, g,* θ_{Res}*) mit*

- $res \in \mathcal{R}$ *der vom Knoten repräsentierten* Ressource,
- $g \in \mathbb{R}^+$ *der Menge an* gebundenen, *d.h. von einem gerade ausführenden Kompetenzmodul gesperrten,* Ressourceeinheiten *und*
- $\theta_{Res} \in]0..\theta]$ *der ressourcenspezifischen* Aktivierungsschwelle *(wobei* θ *die globale Aktivierungsschwelle des Netzwerks ist (siehe unten)).*

Definition 3 *Ein* Kompetenzmodul *besteht aus einem Tupel (*Pre, b, Post, Res, a*) mit*

- Pre $\in \mathcal{L}^\wedge$ *der* Vorbedingung *und* $e = \tau_P(Pre, s)$ *der* Ausführbarkeit*;*
- b *dem* Verhalten, *das ausgeführt wird, wenn das Kompetenzmodul zur Ausführung selektiert wird;*
- Post *einer Menge von Paaren (Eff, ex), wobei* $Eff \in \mathcal{P}$ *ein nach der Ausführung des Verhaltens erwarteter* Effekt *ist und* $ex_j = P(Eff|Pre)$ *die* Erwartungswahrscheinlichkeit $\in [0..1]$ *angibt, mit der die Proposition Eff gegeben Pre wahr wird,*

- Res *einer Menge vom Verhalten b verwendeter* Ressourcen $res \in \mathcal{R}$, *wobei* $\tau_U(k, res, s)$ *die situationsabhängig erwartete Menge an benötigten Ressourceeinheiten des Verhaltens angibt,*
- *a der* Aktivierung $\in \mathbb{R}$, *die die Wünschbarkeit des Verhaltens angibt, wobei* a_g *ein Vektor von Aktivierungen* a_{g_i} *ist, die das Modul direkt oder indirekt von einem Ziel* g_i *empfängt.*

Die Effektwahrscheinlichkeiten ex_i geben die bedingten Wahrscheinlichkeiten an, mit der ein Effekt eintritt, gegeben die Vorbedingung ist erfüllt. Hat ein Verhalten in unterschiedlichen Situationen verschiedene Effekte, kann dem Rechnung getragen werden, indem für jede Situation ein eigenes Kompetenzmodul mit demselben Verhalten eingeführt wird.

Definition 4 *Ein* erweitertes Verhaltensnetzwerk *EBN besteht aus einem Tupel* $(\mathcal{G}, \mathcal{M}, \mathcal{U}, \Pi)$, *wobei* $\mathcal{G}$ *eine endliche Menge von Zielen,* $\mathcal{M}$ *eine endliche Menge von Kompetenzmodulen,* $\mathcal{U}$ *eine endliche Menge von Ressourceknoten und* Π *eine Menge von* Parametern *ist, die den Aktivierungsaustausch und die Verhaltensauswahl steuern. Im Unterschied zu MASM-Netzwerken wurde die Anzahl und der Gültigkeitsbereich der Parameter eingeschränkt:*

- $\gamma \in [0..1[$, *Aktivierung durch Ziele und Nachfolger,*
- $\delta \in [0..1[$, *Hemmung durch Erhaltungsziele und Konfliktoren,*
- $\beta \in [0..1[$, *Trägheit der Aktivierung,*
- $\theta \in]0..\hat{a}]$, *Aktivierungsschwelle mit* $\hat{a}$ *der maximalen Aktivierung eines Moduls,*

3.2 Aktivierungsmechanismus

Die Entscheidung für ein Verhalten soll sowohl von der aktuellen Situation, als auch vom erwarteten Nutzen des Verhaltens abhängig sein. Der erwartete Nutzen eines Verhaltens wird bei erweiterten Verhaltensnetzwerken durch Aktivierungsausbreitung berechnet. Dazu sind die Kompetenzmodule wie in Kapitel 2.2 beschrieben untereinander und mit den Zielen verbunden und tauschen über diese Verbindungen Aktivierung aus. Im Unterschied zu MASM-Netzwerken gibt die Aktivierung eines Kompetenzmoduls durch Ziele in erweiterten Verhaltensnetzwerken den erwarteten Nutzen des zugehörigen Verhaltens an. Sie hängt von der Wahrscheinlichkeit des Beitrags eines Verhaltens zu einem Ziel sowie vom Nutzen des Ziels ab.

Ein Kompetenzmodul k erhält Aktivierung von einem Ziel g_i zum Zeitpunkt t (vgl. Gl. 2.2)

$$a_{kg_i}^{t\,\prime} = \gamma \cdot u(\iota_{g_i}, r_{g_i}^t) \cdot \nu_\gamma(p_j) \cdot ex_j \;, \tag{3.1}$$

falls das Modul einen Effekt p_j mit Erwartungswahrscheinlichkeit ex_j hat, der Teil der Zielbedingung *GCon* ist und beide entweder negiert oder nicht negiert sind. Das Kompetenzmodul ist also in der Lage, Teile der Zielbedingung

zu erfüllen. $u(\iota_{g_i}, r^t_{g_i})$ ist dabei die Nutzenfunktion, die den von Wichtigkeit ι und Relevanz r bestimmten Wert eines Ziels g_i auf einen normierten Nutzen abbildet (siehe Kapitel 3.2.1). Die Funktion ν_γ gibt an, wie dieser Nutzen des Ziels auf die einzelnen Propositionen der Zielbedingung bei aktivierenden Verbindungen verteilt wird. Verwendet wurde $\nu_\gamma(p_j) = \frac{1-\tau_P(p_j,s)}{max(1,\sum_l(1-\tau_P(p_l,s)))}$, mit p_l einer Proposition der Zielbedingung. Der Nutzen einer Proposition p_j der Zielbedingung ist also um so größer, je mehr Propositionen bereits erfüllt sind, und je weniger die Proposition selbst erfüllt ist (siehe Kapitel 3.2.2). Die Aktivierung $a^{t\,\prime}_{kg_i}$ gibt den erwarteten Nutzen der Ausführung des Kompetenzmoduls k bezüglich des Ziels g_i an mit ex_j der Erwartungswahrscheinlichkeit und u dem Nutzen des Ziels. Die Stärke der Aktivierung (im Vergleich zur Hemmung) wird von $\gamma \in [0..1[$ gesteuert. Die Aktivierung, die ein Kompetenzmodul über eine Zielverbindung erhalten kann liegt damit im Intervall $[0..1[$.

Die Hemmung eines Moduls k durch ein Erhaltungsziel g_i beträgt (vgl. Gl. 2.3)

$$a^{t\,\prime\prime}_{kg_i} = -\delta \cdot u(\iota_{g_i}, r^t_{g_i}) \cdot \nu_\delta(p_j) \cdot ex_j \ , \tag{3.2}$$

falls das Modul einen Effekt p_j mit Erwartungswahrscheinlichkeit ex_j hat, der Teil der Zielbedingung *GCon* ist und genau eine beider Propositionen negiert ist. Das Kompetenzmodul kann also potentiell die Zielerreichung verhindern oder, falls die Zielbedingung bereits erfüllt ist, diese wieder zerstören. Auch hier sorgt ν_δ dafür, daß die Hemmung einer erfüllten Proposition um so stärker ist, je mehr Propositionen der Zielbedingung erfüllt sind. Im Unterschied zu ν_γ sorgt $\nu_\delta(p_j) = \frac{\tau_P(p_j,s)}{max(1,\sum_l(1-\tau_P(p_l,s)))}$ dafür, daß der Einfluß einer Proposition um so stärker ist, je mehr die Proposition erfüllt ist (siehe Kapitel 3.2.2). Die Hemmung (negative Aktivierung), die ein Kompetenzmodul über eine Erhaltungszielverbindung erhalten kann, liegt im Intervall $]-1..0]$. Mit Hilfe von Hemmung können die unerwünschten Folgen eines Verhaltens berücksichtigt werden.

Während der Nutzen von Zielen durch Wichtigkeit und Relevanzbedingung direkt angebbar ist, muß der Nutzen von Propositionen, die nicht Teil einer Zielbedingung sind, indirekt berechnet werden. Der Nutzen einer nicht erfüllten Vorbedingungsproposition eines Kompetenzmoduls wird aus der Aktivierung des Kompetenzmoduls sowie aus dem aktuellen Wahrheitswert der Proposition berechnet. Nicht erfüllte Vorbedingungspropositionen werden zu einem impliziten Unterziel im Verhaltensnetzwerk, dessen Nutzen um so höher ist, je höher der erwartete Nutzen des Kompetenzmoduls selbst ist. Kompetenzmodule, die diese Vorbedingungen wahr machen können, werden mit dem Modul verknüpft.

Ein Kompetenzmodul erhält Aktivierung von einem Nachfolgermodul, wenn es einen Effekt p_{succ} mit Erwartungswahrscheinlichkeit ex_j hat, der Teil der Vorbedingung des Nachfolgermoduls ist und beide entweder negiert oder nicht negiert sind. Ein Kompetenzmodul ist also ein Nachfolger eines anderen, wenn dessen Vorbedingung durch das andere Kompetenzmodul erfüllt werden kann. Die Aktivierung wird für jede Zielaktivierung $a_{succ\,g_i}$ des Nachfolgermoduls separat berechnet und beträgt (vgl. Gl. 2.5)

$$a^{t\,\prime\prime\prime}_{kg_i} = \gamma \cdot \sigma(a^{t-1}_{succ\,g_i}) \cdot \nu_\gamma(p_{succ}) \cdot ex_j \ , \tag{3.3}$$

wobei σ: $\mathbb{R} \rightarrow [0..1[$ die Transferfunktion für die Aktivierung angibt, also den erwarteten Nutzen des Kompetenzmoduls auf den Nutzen der nicht erfüllten Vorbedingungsproposition abbildet. Benutzt wurde $\sigma(x) = (1 + e^{\kappa(\mu - x)})^{-1}$ [Goetz, 1997]. Wieder sorgt ν_γ dafür, daß um so mehr Aktivierung an Module geschickt wird, die diese Vorbedingung wahr machen können, je weniger die Proposition des Nachfolgermoduls im Zustand s erfüllt ist. Die nicht erfüllte Vorbedingung wird zu einem Unterziel mit dem Nutzen $\sigma(a^{t-1}_{succ\, g_i}) \cdot \nu_\gamma(p_{succ})$. Die separate Berechnung der Aktivierung für jedes Ziel g_i ist notwendig, da der Zusammenfluß von Aktivierung unterschiedlicher Vorgängerknoten, die von ein und demselben Ziel stammt, anders behandelt wird als der Zusammenfluß von Aktivierung unterschiedlicher Ziele (siehe Kapitel 3.2.3).

Schließlich wird ein Kompetenzmodul durch ein Konfliktormodul gehemmt (vgl. Gl. 2.6)

$$a^{t\,''''}_{kg_i} = -\delta \cdot \sigma(a^{t-1}_{conf\, g_i}) \cdot \nu_\delta(p_{conf}) \cdot ex_j \ , \tag{3.4}$$

falls das Modul einen Effekt p_{conf} hat, der Teil der Vorbedingung des Konfliktormoduls ist und genau eine von beiden Propositionen negiert ist. Ein Kompetenzmodul ist also ein Konfliktor eines anderen, wenn die Vorbedingungen des Konfliktors durch das andere Kompetenzmoduls zerstört werden können. $a^{t-1}_{conf\, g_i}$ ist die Aktivierung, die das Konfliktormodul direkt oder indirekt von Ziel g_i zum Zeitpunkt $t-1$ erhalten hat.

Die Aktivierung, die ein Kompetenzmodul von verschiedenen Propositionen desselben Nachfolgerknotens erhält, wird vom Kompetenzmodul summiert. Je mehr Vorbedingungen eines Nachfolgers ein Kompetenzmodul also wahr machen kann, desto mehr Aktivierung erhält es. Die Aktivierung a_{kg_i}, die ein Kompetenzmodul k direkt oder indirekt von verschiedenen Nachfolgerknoten von Ziel g_i erhält, wird von k separat gespeichert und durch die Verbindung mit dem maximalen Betrag an Aktivierung festgelegt:

$$a^t_{kg_i} = abs\max(a^{t\,'}_{kg_i}, a^{t\,''}_{kg_i}, a^{t\,'''}_{kg_i}, a^{t\,''''}_{kg_i}) \tag{3.5}$$

Ein Kompetenzmodul berücksichtigt also nur den stärksten Pfad zu einem Ziel. Verschiedene eingehende Verbindungen können so als Alternativen zur Erreichung eines Ziels erkannt werden. Unter der Annahme, daß die beste der Alternativen verfolgt wird, trägt nur die stärkste Verbindung zum maximal erreichbaren Nutzen bei (siehe Kapitel 3.2.3).

Die Aktivierung, die ein Kompetenzmodul von unterschiedlichen Zielen erhält, wird vom Kompetenzmodul summiert, da jeder Beitrag zu verschiedenen Zielen den maximal erreichbaren Nutzen erhöht. Somit beträgt die Gesamtaktivierung eines Moduls k schließlich

$$a^t_k = \beta a^{t-1}_k + \sum_i a^t_{kg_i} \ , \tag{3.6}$$

wobei β die Trägheit der Aktivierung und damit die Trägheit des Verhaltens steuert.

Die Ausgangsaktivierung ist $a^0_k = 0$. a^1_k entspricht der Aktivierung, die ein Kompetenzmodul direkt von den Zielen erhält, da die Aktivierung bzw. Hemmung durch Nachfolger- und Konfliktormodule noch vernachlässigbar ist

($\sigma(a_k^0) = \sigma(0) \approx 0$) (Gl. 3.3, 3.4). a_k^2 berücksichtigt bereits den indirekten Einfluß der Ziele auf Kompetenzmodule über ein anderes Kompetenzmodul. Es werden also schon Verhaltensfolgen der Länge zwei berücksichtigt, die zur Erreichung eines Ziels führen. Jede weitere Aktivierungsberechnung erlaubt die Berücksichtigung einer um eins längeren Verhaltensfolge. Je länger der Aktivierungsaustausch also stattfindet, desto weiter kann in die Zukunft geplant werden, desto größer ist die Antizipationsweite. Aufgabe der Verhaltensauswahl (Kapitel 3.3) ist es, einen Kompromiß zwischen ausreichendem Aktivierungsaustausch für langfristige Planung und schnellem, situationsangepaßtem Handeln zu finden.

3.2.1 Nutzenfunktion

Die Nutzenfunktion u gibt den subjektiven Nutzen eines Ziels an. Da die Axiome der Nutzentheorie die Skalierung der Nutzenfunktion nicht festlegen, wird der Nutzen eines Ziels gewöhnlich als normalisierter Nutzen angegeben. Der minimale Nutzen $u_\perp$ wird auf den Wert 0, der maximale Nutzen $u_\top$ auf den Wert 1 abgebildet [Russel und Norvig, 1995]. Wird dazwischen linear interpoliert, entsteht ein risiko-neutrales Verhalten. Nicht-lineare Nutzenfunktionen führen zu risiko-geneigtem bzw. risiko-aversivem Verhalten (siehe Kapitel 6.1).

Die verwendete Nutzenfunktion $u(\iota_{g_i}, r_{g_i}^t) = \iota_{g_i} \cdot r_{g_i}$ verknüpft die statische Wichtigkeit ι_{g_i} und die dynamische Relevanz r_{g_i} des Ziels g_i. Der Nutzen eines unwichtigen ($\iota_{g_i} = 0$) oder nicht relevanten Ziels ($r_{g_i} = 0$) ist immer Null. Da sowohl die Relevanz $r_{g_i} = \tau_P(RCon, s)$ als auch die Wichtigkeit ι_{g_i} eines Ziels im Intervall [0..1] liegt, gilt $0 \leq u \leq 1$.

Durch die Relevanzbedingung eines Ziels können, anders als etwa bei entscheidungstheoretischem Planen in Markov-Entscheidungsprozessen, auch dynamische Nutzenfunktionen repräsentiert werden. Der Nutzen eines Ziels kann sich beispielsweise ändern, wenn der Termin zur Erreichung des Ziels verstrichen ist. Durch eine Relevanzbedingung, deren Wahrheitswert τ_P entsprechend nach dem Verstreichen eines Termins abfällt, kann solchen dynamischen Nutzenfunktionen Rechnung getragen werden (siehe auch Kapitel 6.1.3).

3.2.2 Verteilungsfunktion

Die Verteilungsfunktion ν gibt an, wie der Nutzen eines Ziels oder eines Kompetenzmoduls auf die einzelnen Propositionen der Ziel- bzw. Vorbedingung verteilt wird. ν_γ sorgt bei aktivierenden Verbindungen dafür, daß die Aktivierung um so höher ist, je weniger die Proposition erfüllt ist ($1 - \tau_P(p, s)$). ν_δ sorgt im Gegensatz dazu bei hemmenden Verbindungen dafür, daß die Aktivierung um so höher ist, je mehr die Proposition bereits erfüllt ist ($\tau_P(p, s)$). Beide Funktionen teilen die Aktivierung durch die Anzahl nicht erfüllter Vorbedingungen $\sum_l(1 - \tau_P(p_l, s))$. Dadurch ist der Nutzen einer Proposition um so größer, je mehr Vorbedingungen eines Ziels oder Kompetenzmoduls bereits erfüllt sind. Um die maximale Aktivierung einer Proposition auf $(1-\tau_P(p, s))$ bzw. $(\tau_P(p, s))$ zu beschränken, wird die untere Grenze der Anzahl nicht erfüllter Vorbedingungen eins gesetzt ($max(1, \sum)$). Dadurch wird verhindert, daß der Nutzen einer

Proposition größer als der Nutzen des zugehörigen Ziels bzw. Kompetenzmoduls ist.

Die Verteilungsfunktion ist nicht mit dem Fan-Effekt in MASM-Netzwerken zu verwechseln. Während letzterer dafür sorgt, daß die Aktivierung eines Knotens durch alle ausgehenden Verbindungen geteilt wird, sorgt die Verteilungsfunktion in erweiterten Verhaltensnetzwerken dafür, daß die Aktivierung durch die Anzahl nicht erfüllter Bedingungspropositionen geteilt wird. Wieviele Verbindungen zu einer konkreten Proposition existieren, spielt jedoch keine Rolle. Der erwartete Nutzen eines Verhaltens ist so, anders als bei MASM-Netzwerken, unabhängig davon, wieviele andere Verhalten dieselbe Proposition wahr machen können.

3.2.3 Goaltracking

Im Unterschied zu MASM-Netzwerken wird die Aktivierung von verschiedenen Vorgängerknoten von einem Kompetenzmodul nicht summiert, sondern es berücksichtigt deren Herkunft. Aktivierung von verschiedenen Vorgängerknoten wird nur summiert, wenn sie von unterschiedlichen Zielen stammt (Gl. 3.6). Bei Aktivierung, die vom selben Ziel stammt, wird nur die Verbindung mit dem höchsten Aktivierungsbetrag berücksichtigt (Gl. 3.5), d.h. entweder der stärkste hemmende oder aktivierende Einfluß jedes Ziels wird berücksichtigt. Diese als Goaltracking bezeichnete Methode der Aktivierungsweitergabe sorgt dafür, daß die Aktivierung eines Kompetenzmoduls durch ein Ziel den erwarteten Nutzen des Verhaltens angibt, und die Aktivierungsweitergabe den maximal erreichbaren Nutzen des Verhaltens berücksichtigt.

Für Kompetenzmodule, die direkt mit Zielen verbunden sind, läßt sich der erwartete Nutzen unmittelbar berechnen, da sowohl die Erwartungswahrscheinlichkeit ex_j für das Erreichen eines Ziels als auch der Nutzen des Ziels $u(\iota_{g_i}, r^t_{g_i})$ bekannt sind. Für Effekte, die nicht Teil einer Zielbedingung sind, ist jedoch der Nutzen nicht explizit gegeben. Mit Hilfe der Aktivierungsweitergabe wird versucht, auf den Nutzen von Propositionen indirekt aus dem erwarteten Nutzen von Verhalten zu schließen. Der erwartete Nutzen hängt dabei von der Anzahl Aktionen ab, die ausgeführt werden können. Der erwartete Nutzen von 'action3' in Abb. 3.1 ist beispielsweise 0, falls nur eine Aktion ausgeführt werden kann, da die Effekte von 'action3' nicht direkt ein Ziel erreichen können, also keinen direkten Nutzen haben. Erst wenn mehr als eine Aktion ausgeführt werden kann, ist der erwartete Nutzen größer als 0 und geht bei unendlicher Anzahl Aktionsausführungen gegen die Summe der Nutzen der erreichbaren Ziele, da die Wahrscheinlichkeit, die Ziele zu erreichen, dann 1 ist.

Definition 5 *Zwischen zwei Knoten k_1, k_2 eines erweiterten Verhaltensnetzwerks existiert ein* gerichteter Pfad *$Pfad(k_1, k_2)$, falls es eine Verbindung von k_1 nach k_2 gibt oder falls es einen Knoten k_3 gibt, so daß gilt: $Pfad(k_1, k_3)$ und $Pfad(k_3, k_2)$.*

Der maximal erreichbare Nutzen eines Kompetenzmoduls ist dann

$$mru_{k_i} = \sum_{j, \exists Pfad(g_j, k_i)} u(g_j). \tag{3.7}$$

Der Aktivierungsmechanismus berücksichtigt den maximal erreichbaren Nutzen, indem der erwartete Nutzen unterschiedlicher Ziele addiert wird (Gl. 3.6) jedoch nur ein Pfad zu einem Ziel berücksichtigt wird (Gl. 3.5).

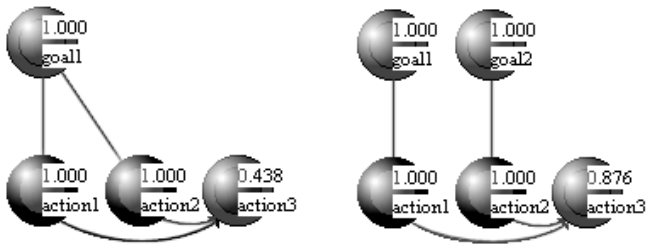

Abbildung 3.1: Vermeidung von Aktivierungszusammenfluß in 'action3' (links) und erwünschter Aktivierungszusammenfluß bei mehreren Zielen durch Goaltracking (rechts).

Goaltracking verhindert, daß Ziele mit mehreren alternativen, zielerfüllenden Kompetenzmodulen benachteiligt werden (siehe Kapitel 2.5.2), da an jedes dieser Module die volle Aktivierung weitergegeben wird. Gleichzeitig wird verhindert, daß Aktivierung vom selben Ziel über mehrere Kompetenzmodule in einem Modul zusammenfließt (Abb. 3.1), wie es beim einfachen Weglassen des Fan-Effekts der Fall wäre [Tyrrell, 1994].

Weiterhin verhindert Goaltracking Aktivierungsrückkopplung, da nur in seltenen Fällen (siehe folgendes Kapitel) eine eingehende Verbindung eines Zykels die stärkste eingehende Verbindung eines Kompetenzmoduls ist. Dadurch tritt keine Bevorzugung appetitiver Knoten auf, wie es in MASM-Netzwerken der Fall ist (Kapitel 2.5.1).

Goaltracking kann lokal, d.h. basierend auf der Information eines Kompetenzmoduls und dessen Verbindungen, durchgeführt werden. Die Möglichkeit zur parallelen Berechnung der Aktivierung jedes Kompetenzmoduls bleibt beibehalten.

Die Kosten von Goaltracking sind ein erhöhter Zeit- und Platzbedarf von $O(|\mathcal{M}|^2|\mathcal{G}|)$ Zeitkomplexität für die (serielle) Berechnung der Aktivierung aller Kompetenzmodule gegenüber $O(|\mathcal{M}|^2)$ bei MASM und $O(|\mathcal{P}| + |\mathcal{G}|)$ Platzaufwand für die Repräsentation eines Kompetenzmoduls gegenüber $O(|\mathcal{P}|)$ bei MASM.

Goaltracking behebt Probleme des Aktivierungsmechanismus, der für zielgerichtetes Handeln sorgt. Deshalb ist zu erwarten, daß Goaltracking vor allem in solchen Domänen eine Verbesserung der Qualität der Handlungskontrolle bewirkt, in denen zielgerichtetes Handeln besonders wichtig ist. Da Goaltracking Aktivierungsrückkopplung und Aktivierungszusammenfluß verhindert, sollten weiterhin insbesondere große Verhaltensnetzwerke mit vielen Aktivierungsverbindungen von Goaltracking profitieren, da diese Probleme besonderes in solchen Netzwerken zu erwarten sind.

3.2.4 Stabilität

Eine wichtige Eigenschaft des Aktivierungsmechanismus ist es, einen stabilen Zustand zu erreichen. Das ist nicht der Fall, falls die Aktivierung eines Kompetenzmoduls über alle Grenzen wachsen kann, oder falls es Oszillationen von Aktivierung innerhalb des Netzwerks geben kann. Zwar funktioniert die Verhaltensauswahl (siehe Kapitel 3.3) auch bei oszillierender Aktivierung von Kompetenzmodulen, das Verhalten wird aber potentiell instabiler, d.h. es können mehr Verhaltenswechsel auftreten, insbesondere dann, wenn die Aktivierung des gerade ausgeführten Moduls oszilliert. Im allgemeinen kann man davon ausgehen, daß stabileres Verhalten bessere Ergebnisse liefert, wie auch empirische Untersuchungen zeigen (siehe Kapitel 5).

Lemma 6 *Der beschriebene Algorithmus zur Aktivierungsausbreitung in erweiterten Verhaltensnetzwerken verhindert Aktivierungsrückkopplung für* $t \to \infty$.

Beweis. Es ist möglich, daß Kompetenzmodule über Zykel Aktivierung von sich selbst erhalten (siehe Abb. 3.2). Diese Aktivierung $a^t_{self\,g_i}$ geht aber gegen Null für $t \to \infty$. Das gilt, weil $|a^t_{self\,g_i}| < \gamma^{n\zeta}\delta^{m\zeta}$ mit $\zeta = \frac{t}{n+m}$ (Gl. 3.3, 3.4 und $\tau_P(p,s), ex_j, \sigma(a) \leq 1$). n ist dabei die Anzahl aktivierender Verbindungen, m die Anzahl hemmender Verbindungen innerhalb des Zykels. Bei jeder Weitergabe von Aktivierung wird diese verringert, da $\gamma, \delta < 1$. Für $t \to \infty$ geht entweder $\gamma^{n\zeta}$ (für $n > 0$) oder $\delta^{m\zeta}$ (für $m > 0$) und damit $a_{self}g^t_i$ gegen Null. Für $m = n = 0$ ist das Kompetenzmodul nicht Teil eines Zykels. □

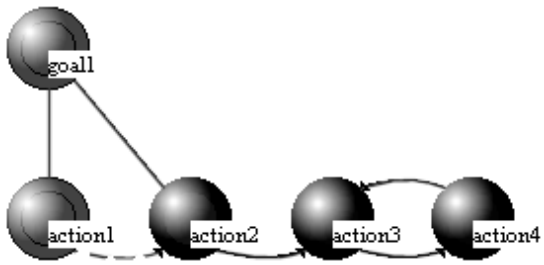

Abbildung 3.2: Beispiel für ein Netzwerk, bei dem Kompetenzmodule (action3 und action4) Aktivierung über einen Zykel von sich selbst erhalten. Die Aktivierung stammt ursprünglich von Modul 'action2', das seinerseits die Aktivierung vom Ziel erhalten hat, seit dem zweiten Aktivierungszyklus aber von Modul 'action1' gehemmt wird.

Theorem 7 *Erweiterte Verhaltensnetzwerke erreichen immer einen stabilen Aktivierungszustand (bei gleichbleibender Situation).*

Beweis. Die Aktivierung von Zielen a_{kg_i} wird von Kompetenzmodulen für jedes Ziel separat berechnet und gespeichert. Daher kann man jedes Ziel separat behandeln und die Untergraphen des Netzwerks mit je einem Ziel betrachten. Die

Knoten in einem solchen Untergraphen bestehen aus dem Ziel und den Kompetenzmodulen und werden in zwei Mengen aufgeteilt: V^0 enthält die Knoten, die einen stabilen Aktivierungseingang erhalten, V enthält alle anderen Knoten des Untergraphen. Zunächst enthält V^0 nur das Ziel. Nach einem Aktivierungszyklus erhält das Kompetenzmodul mit der stärksten Verbindung zum Ziel (im Sinn von maximalem Aktivierungsbetrag) einen konstanten Betrag an Aktivierung vom Ziel und kann von V entfernt und in V^0 eingefügt werden. Das gilt, weil Aktivierung durch Aktivierungsweitergabe immer abnimmt (Beweis von Lemma 6) und daher das Kompetenzmodul über keinen anderen Pfad mehr Aktivierung vom Ziel erhalten kann. Dies kann nacheinander für alle Knoten in V durchgeführt werden, auch wenn die Aktivierung dieser Knoten mehr als einen Aktivierungszyklus benötigen kann, um einen stabilen Zustand zu erreichen. Das ist dann der Fall, wenn Aktivierungsrückkopplung stattfindet. Diese Aktivierungsrückkopplung geht jedoch gegen Null für $t \to \infty$ (Lemma 6). Die Aktivierung eines Kompetenzmoduls m beträgt dann $a_m = (1-\beta)^{-1} \sum_n a_{mg_n}$ (Gl. 3.6) und $\Delta a_m = 0$. □

Die maximale ($\hat{a}$) und minimale ($\check{a}$) Aktivierung, die ein Kompetenzmodul erreichen kann beträgt

$$\hat{a} = \frac{|\mathcal{G}|}{1-\beta} \qquad \check{a} = \frac{-|\mathcal{G}|}{1-\beta}\,, \tag{3.8}$$

hängt also nur von der Anzahl der Ziele und von β ab und nicht von der Anzahl Kompetenzmodule.

3.2.5 Situationsaktivierung

MASM-Netzwerke enthalten zwei weitere Arten von Verknüpfungen zu den Vorbedingungen, über die ein Kompetenzmodul Aktivierung erhält. Zum einen erhält jede erfüllte Vorbedingung von der entsprechenden Situationsproposition Aktivierung (Situationsverbindung), zum anderen erhält jede nicht erfüllte Vorbedingung Aktivierung von ausführbaren Kompetenzmodulen, die diese Vorbedingung wahr machen können (Vorgänger-Verbindung). Diese Verbindungen sollen für die Reaktivität des Agenten sorgen [Maes, 1990b], indem Module mit einem größeren Anteil erfüllter Vorbedingungen mehr Aktivierung erhalten.

Im folgenden wird der Einfluß von Situationsaktivierung auf ausführbare und auf nicht ausführbare Kompetenzmodule in MASM-Netzwerken unterschieden. Durch den Fan-Effekt erhalten diejenigen ausführbaren Kompetenzmodule mehr Aktivierung, deren Vorbedingungen seltener in Vorbedingungen anderer Kompetenzmodule auftreten, da die Aktivierung von Situationspropositionen durch die Anzahl empfangender Kompetenzmodule geteilt wird (Gl. 2.1). Wie bei Zielpropositionen ist es aber auch hier fraglich, warum Module mit einer im Verhaltensnetzwerk häufig vorkommenden Vorbedingung benachteiligt werden sollen. Verzichtet man, wie vorgeschlagen, auf den Fan-Effekt, erhalten alle ausführbaren Kompetenzmodule dieselbe Aktivierung ϕ (Gl. 2.1). Die Situationsaktivierung hat dann keinen direkten Einfluß auf die Auswahl des auszuführenden Moduls.

Der indirekte Einfluß auf die Verhaltensauswahl durch die höhere Aktivierung nicht ausführbarer Kompetenzmodule mit einigen erfüllten Vorbedingungen zeigte in empirischen Studien [Goetz, 1997; Dorer, 1999a] keine Verbesserung der Verhaltensauswahl. Die Variation des Parameters für die Situationsaktivierung ϕ zeigte in beiden Studien die beste Performanz für $\phi = 0$.

Daher ist Situationsaktivierung in erweiterten Verhaltensnetzwerken nicht vorhanden. Einzige Quellen für Aktivierung sind die Ziele. Der Einfluß der Situation auf die Verhaltensauswahl in erweiterten Verhaltensnetzwerken ist unter anderem durch die Ausführbarkeit von Kompetenzmodulen gegeben.

3.3 Verhaltensauswahl

Zur Koordination gleichzeitig auszuführender Verhalten sind die Kompetenzmodule in $\mathcal{M}$ mit den Ressourceknoten in $\mathcal{U}$ verbunden. Ein Kompetenzmodul besitzt für jede benötigte Ressource res eine *Ressourcenverbindung* zum zugehörigen Ressourceknoten. Diese Verbindungen ermöglichen dem Kompetenzmodul die Verfügbarkeit einer Ressource zu kontrollieren. Die Verhaltensauswahl kann dadurch lokal in den Kompetenzmodulen erfolgen. Sie wird für jedes Kompetenzmodul k zyklisch in folgenden Schritten durchgeführt(vgl. 2.3):

1. Berechne die Aktivierung a_k^t (Gl. 3.6) und Ausführbarkeit $e = \tau_P(Pred, s)$ des Kompetenzmoduls und verknüpfe sie mit einer monoton wachsenden Funktion $h : \mathbb{R} \times [0..1] \rightarrow \mathbb{R}$, die den *Ausführungswert* des Moduls angibt. Damit nur ausführbare Verhalten ausgeführt werden, muß h so gewählt werden, daß $h(a, 0) = 0$ ist. Für die empirischen Untersuchungen im Rahmen dieser Arbeit wurde $h(x, y) = x \cdot y$ verwendet.

2. Für jede vom Kompetenzmodul k verwendete Ressource res, beginnend bei der zuletzt nicht verfügbaren Ressource

   (a) Prüfe, ob der Ausführungswert h größer als die Aktivierungsschwelle θ_{Res_i} des zugehörigen Ressourceknotens ist.

   (b) Prüfe, ob genügend Einheiten der verwendeten Ressource vorhanden sind, d.h. prüfe, ob $\tau_U(k, res, s) \leq \tau_R(res, s)$. Wenn ja, binde die Ressource an das Verhaltensmodul, d.h. erhöhe die Anzahl gebundener Ressourceeinheiten g des zugehörigen Ressourceknotens um die Anzahl vom Verhalten benötigter Ressourceeinheiten τ_U.

3. Falls alle Tests in 2. erfolgreich sind

   (a) Führe das zugehörige Verhalten b aus.

   (b) Setze die Aktivierungsschwellen aller verwendeten Ressourcen θ_{Res_i} auf den ursprünglichen Wert θ

4. Gib alle gebundenen Ressourcen frei, d.h. erniedrige die Menge der gebundenen Ressourceeinheiten g des zugehörigen Ressourceknotens um die zuvor gebundenen Ressourceeinheiten τ_U.

5. Wiederhole ab 1.

Die Aktivierungsschwelle θ_{Res_i} in den Ressourceknoten ist einem zeitlichen Zerfall unterworfen. Verwendet wurde ein linearer Zerfall, wobei gilt: $\theta_{Res_i} > 0$, d.h. der lineare Zerfall endet bei einem Wert größer null. Dadurch wird gewährleistet, daß nur Kompetenzmodule mit einem Ausführungswert größer null, also nur ausführbare und aktivierte Module, ausgeführt werden.

Die Prüfung auf Vorhandensein einer Ressource in Punkt 2 sollte bei der zuletzt nicht verfügbaren Ressource beginnen, da die Wahrscheinlichkeit für das Scheitern der Prüfung bei dieser Ressource besonders hoch ist. Dadurch ist einerseits die durchschnittlich benötigte Zeit für die Prüfung der Ressourcen kürzer. Zum anderen sind andere vom Verhalten benötigte Ressourcen kürzer, bzw. gar nicht gesperrt und sind so für die übrigen Kompetenzmodule schneller zugreifbar.

3.3.1 Parallele Ausführung von Verhalten

Wie in Kapitel 2.5.3 ausgeführt, ist die Verhaltensauswahl bei MASM-Netzwerken auf die Selektion und Ausführung genau eines Verhaltens je Zeiteinheit beschränkt. Der Mensch ist jedoch in der Lage, mehrere Verhalten gleichzeitig auszuführen, solange die Verhalten nicht dieselben physischen oder kognitiven Ressourcen verwenden [Shaffer, 1975; Navon und Gopher, 1979; Norman und Shallice, 1986]. Beispielsweise kann eine gut geübte Stenotypistin gleichzeitig einen Text tippen, den sie liest und einen anderen Text laut nachsprechen, den sie hört [Shaffer, 1975].

Um in erweiterten Verhaltensnetzwerken die gleichzeitige Ausführung von Verhalten zu erlauben, wurde der Begriff der Ressource eingeführt. Während Norman und Bobrow [1975] unter Ressourcen interne Verarbeitungskapazitäten verstehen, wird der Begriff der Ressource in dieser Arbeit weiter gefaßt und kann sich sowohl auf (interne) Verarbeitungsressourcen als auch auf (externe) physische Ressourcen beziehen. Die Information jedes Kompetenzmoduls k wird um die vom Verhalten verwendeten Ressourcen res_i erweitert. Die benötigte Menge einer Ressource $\tau_U(k, res_i, s)$ kann dabei von der aktuellen Situation abhängen, was besonders bei verbrauchenden Ressourcen (z.B. Energie, Treibstoff, ...) der Fall ist. Das Verhalten 'zum Ball rennen' im Fußball benötigt beispielsweise entsprechend mehr Ausdauer, je weiter der Ball vom Agenten entfernt ist. Bei wiederverwendbaren Ressourcen (z.B. Werkzeug, Prozessor, ...) ist die Menge meist unabhängig von der Situation konstant.

Für die zeitgleiche Ausführung von Verhalten muß der Zugriff auf Ressourcen koordiniert werden. Im Gegensatz zu zentralen Ressourcenmodellen, wie sie etwa von Norman und Bobrow [1975] beschrieben werden, lehnt sich das in dieser Arbeit verwendete Ressourcenmodell an die Annahme multipler Ressourcen an [Navon und Gopher, 1979]. Entsprechend werden verschiedene Ressourcenpools unterschieden und durch jeweils eigene Ressourceknoten repräsentiert. Die zusätzlich in das Netzwerk eingeführten Ressourceknoten sorgen für die Koordination der Verhalten untereinander. Kommt ein Verhalten zur Ausführung, was nur möglich ist, wenn alle benötigten Ressourcen vorhanden sind, sperrt es die von ihm benötigten Ressourcen für die Zeit der Ausführung. Andere Verhalten, die dieselbe Ressource verwenden, sind in dieser Zeit nicht durchführbar,

wenn die Ressource nicht in ausreichender Menge vorhanden ist. Verhalten, die keine Ressourcenkonflikte mit anderen, gerade ausführenden Verhalten haben, können zeitgleich ausgeführt werden.

3.3.2 Aktivierungsschwelle

Die Verhaltensauswahl muß dafür sorgen, daß zum einen schnell auf Änderungen in der Situation reagiert werden kann, zum anderen aber auch genügend Zeit in die langfristige Planung investiert wird. Sie muß also entscheiden, ob es sich weiterhin lohnt, Aktivierungsaustausch durchzuführen und damit die Antizipationsweite zu vergrößern (siehe Kapitel 3.2) oder ob ein oder mehrere Kompetenzmodule zur Ausführung gelangen sollen.

Erreicht wird dies dadurch, daß ein Kompetenzmodul bevor es zur Ausführung gelangen kann die Aktivierungsschwelle θ_{Res_i} jeder verwendeten Ressource überschritten haben muß. θ_{Res_i} wird anfänglich und nach der Ausführung eines Verhaltens, das die Ressource R_i benützt, mit dem Wert der globalen Aktivierungsschwelle θ belegt. Je höher θ gewählt wird, desto länger wird in der Regel Aktivierung ausgetauscht und desto größer ist die Antizipationsweite.

Die Zuordnung einer Aktivierungsschwelle zu jeder Ressource ist notwendig, da mit einer einzigen Aktivierungsschwelle weiterhin nur das am meisten aktivierte Kompetenzmodul zur Ausführung gelangen würde. Kompetenzmodule, die parallel ausführbar wären, jedoch eine geringere Aktivierung besitzen, würden nicht ausgeführt, da nach der Ausführung des aktivsten Moduls die Aktivierungsschwelle auf ihren ursprünglichen Wert zurückgesetzt würde. Anschaulich gesprochen wettstreiten die Kompetenzmodule nicht mehr wie bei MASM-Netzwerken um die eine Ressource 'Agent', sondern um die tatsächlich vom jeweiligen Kompetenzmodul benötigten Ressourcen. Es gibt also nicht eine einzige zentrale Ressource sondern multiple Ressourcen. Das hat zur Folge, daß nicht immer das am meisten aktivierte Kompetenzmodul ausgeführt wird. Sind nicht alle Ressourcen des aktivsten Moduls verfügbar, kann auch ein Kompetenzmodul mit geringerer Aktivierung zur Ausführung kommen. Wird hingegen eine Ressource längere Zeit von keinem Verhalten in Anspruch genommen, ist deren Aktivierungsschwelle niedrig. Ein Verhalten, das nur diese Ressource benötigt, kann mit sehr geringer Aktivierung ausgeführt werden, da es (momentan) das einzige ist, das um diese Ressource wettstreitet.

3.3.3 Reellwertige Propositionen

Reale Umgebungen enthalten meist kontinuierliche Zustände. So kann z.B. der Ladungszustand einer Batterie beliebige Werte zwischen 'leer' und 'voll' annehmen. In solchen Umgebungen ist es wünschenswert, den exakten Wert der Proposition direkt bei der Verhaltenskontrolle berücksichtigen zu können, um z.B. Aussagen zu formulieren wie: 'je leerer die Batterie ist, desto dringender muß sie aufgeladen werden'.

Um die aktuelle Situation auch in kontinuierlichen Umgebungen mit kontinuierlichen Zuständen so genau wie möglich erfassen zu können, wurden reellwertige Propositionen (τ_P-Funktion) eingeführt. Die Einführung reellwerti-

ger Propositionen hat Einfluß auf die Ausführbarkeit von Verhalten, auf die Aktivierungsausbreitung innerhalb des Netzwerks, sowie auf die Relevanz von Zielen.

Anders als bei MASM-Netzwerken sind Kompetenzmodule in erweiterten Verhaltensnetzwerken nicht mehr entweder ausführbar oder nicht ausführbar, sondern sie sind zu einem gewissen Grad ausführbar. Daher kann nicht einfach aus der Menge der ausführbaren Module ein Kompetenzmodul zur Ausführung selektiert werden. Es muß vielmehr ein Kompromiß zwischen Aktivierung und Ausführbarkeit gefunden werden. Die Verknüpfung von Aktivierung und Ausführbarkeit zum Ausführungswert h im ersten Schritt der Verhaltensauswahl entspricht einem Kompromiß zwischen dem Nutzen eines Verhaltens und der Wahrscheinlichkeit einer erfolgreichen Ausführung. Der Vorteil ist, daß ein Verhalten mit hohem Nutzen auch dann ausgeführt werden kann, wenn nicht alle Vorbedingungen perfekt erfüllt sind. Im Gegensatz zu den Erwartungswerten der Effekte eines Kompetenzmoduls, die ein Maß für die subjektive Kompetenz des Agenten sind [Dörner, 1999], spiegelt die Ausführbarkeit eines Kompetenzmoduls die Gelegenheit der Situation wider, das zugehörige Verhalten auszuführen. In der motivationspsychologischen Literatur wird in diesem Zusammenhang vom Kompromiß zwischen Wünschbarkeit und Realisierbarkeit gesprochen [Heckhausen, 1989].

Reellwertige Propositionen haben Einfluß auf die Aktivierungsausbreitung, da die Aktivierungsweitergabe von Kompetenzmodulen vom Wahrheitswert der entsprechenden Vorbedingung abhängt. Während in MASM-Netzwerken die Aktivierung von Nachfolgern und die Hemmung von Konfliktoren binären Charakter hat, ist in erweiterten Verhaltensnetzwerken die Aktivierung bzw. Hemmung um so stärker, je weniger bzw. mehr die zugehörige Vorbedingung des Kompetenzmoduls erfüllt ist.

Die Einführung reellwertiger Propositionen erlaubt weiterhin einen graduellen Einfluß der Ziele auf das Verhalten. Da die Relevanzbedingung eines Ziels beliebige Wahrheitswerte zwischen null und eins annehmen kann, ist auch dessen Aktivierung graduell. Je relevanter ein Ziel ist, desto mehr Aktivierung geht von diesem aus. (siehe Kapitel 6.1).

3.3.4 Verhaltensparametrisierung

In MASM-Netzwerken hat der Prozeß der Auswahl eines Verhaltens mittels des Verhaltensnetzwerks lediglich Einfluß darauf, welches der Verhalten ausgeführt wird, nicht aber auf das auszuführende Verhalten selbst. In biologischen Systemen hat die Entschlossenheit der Entscheidung auch Einfluß auf die Ausführung von Verhalten. "Intensität und Ausdauer der Handlung wird von der Volitionsstärke der Zielintention bestimmt" [Heckhausen, 1989]. Daher wäre es wünschenswert, die Entschlossenheit der Entscheidung in die Ausführung eines Verhaltens einfließen zu lassen. So könnte z.B. das 'renne zum Ball' Verhalten eines Fußballagenten von der Entschlossenheit abhängig gemacht werden, dieses Verhalten auszuführen. Je höher der erwartete Nutzen und die Ausführbarkeit des Verhaltens sind, desto mehr lohnt sich der höhere Einsatz von Ressourcen (Ausdauer).

Ein geeignetes Maß für die Entschlossenheit in erweiterten Verhaltensnetzwerken ist der Ausführungswert eines Kompetenzmoduls (Kapitel 3.3). Dieser spiegelt sowohl den erwarteten Nutzen eines Verhaltens für die Zielerreichung als auch die Situationsadäquatheit des Verhaltens wider. Problematisch für die Verhaltensparametrisierung ist jedoch, daß der Ausführungswert eines Kompetenzmoduls von der Anzahl der Ziele im Verhaltensnetzwerk $|\mathcal{G}|$ abhängig ist. Ein parametrisiertes Verhalten sollte aber unabhängig von der konkreten Netzwerkarchitektur sein. Es ist also notwendig, den Ausführungswert in geeigneter Weise zu normieren, d.h. auf einen von der Netzwerkarchitektur unabhängigen Wertebereich abzubilden. Im folgenden werden drei Vorschläge für eine solche Abbildung auf den Wertebereich [0..1] erörtert.

Es liegt nahe, den Ausführungswert durch die Anzahl Ziele $|\mathcal{G}|$ zu teilen und somit unabhängig von dieser zu machen. $|\mathcal{G}|$ ist aber innerhalb eines Kompetenzmoduls nicht verfügbar. Ein Kompetenzmodul kennt lediglich die Anzahl der Ziele, von denen es direkt oder indirekt Aktivierung empfängt. Die Normierung des Ausführungswerts mittels Division durch die Gesamtanzahl Ziele verletzt somit das Lokalitätsprinzip und ist ungeeignet.

Ein Verfahren, das auf lokaler Information eines Kompetenzmoduls beruht, ist es, den minimalen ($\hat{h}$) und maximalen ($\check{h}$) Ausführungswert des Moduls zu speichern und den Verhaltensparameter p als

$$p = \frac{h - \check{h}}{\hat{h} - \check{h}} \tag{3.9}$$

zu berechnen, wobei h der momentane Ausführungswert des Kompetenzmoduls ist. Problematisch ist hierbei, daß diese Art der Normierung anfällig ist gegenüber einzelnen besonders hohen oder niedrigen Ausführungswerten.

Dieses Problem tritt nicht auf, wenn man anstelle der Extremwerte die Verteilung der Ausführungswerte eines Kompetenzmoduls berücksichtigt. Geht man davon aus, daß die Ausführungswerte normalverteilt sind, genügt es, den Mittelwert und dessen Standardabweichung der Ausführungswerte zu berechnen. Die Abbildung des Ausführungswerts auf den Verhaltensparameter p erfolgt dann nach

$$p = \begin{cases} 0 & : \quad h < \mu - k \cdot s \\ \frac{h-(\mu - k \cdot s)}{2k \cdot s} & : \quad \mu - k \cdot s \leq h \leq \mu + k \cdot s \\ 1 & : \quad \mu + k \cdot s < h \end{cases} \tag{3.10}$$

wobei k den Bereich festlegt, der auf das Intervall [0..1] abgebildet wird. Die Berechnung von μ und s erfolgt inkrementell:

$$\mu_{n+1} = \mu_n + \frac{h - \mu_n}{n+1} \quad und \tag{3.11}$$

$$s^2_{n+1} = (n+1) \cdot (\mu_{n+1} - \mu_n)^2 + \frac{(n-1) \cdot s^2}{n} \tag{3.12}$$

In empirischen Untersuchungen (siehe Kapitel 5.3) war dieses Verfahren erfolgreicher als die Normierung mittels Extremwerten. Beide Verfahren zeigten jedoch in der Fußballumgebung eine höhere Erfolgsrate als sie ohne Verhaltensparametrisierung erreicht wurde.

3.3.5 Lokalitätsprinzip

Wie in Kapitel 2.5.3 ausgeführt, verletzt MASM das Lokalitätsprinzip in drei Punkten:

- Division der Aktivierung durch die Anzahl empfangender/sendender Kompetenzmodule.
- Aktivierungsnormierung
- Selektion des am meisten aktivierten Kompetenzmoduls

Die Division der Aktivierung entfällt in erweiterten Verhaltensnetzwerken durch die Einführung des Goaltracking (Kapitel 3.2.3). Alle für das Goaltracking benötigten Informationen sind entweder lokal gespeichert (maximaler Aktivierungsbetrag von jedem Ziel a_{kg_i}) oder sind über verbundene Kompetenzmodule erreichbar (maximaler Aktivierungsbetrag von Nachfolger- und Konfliktormodulen für jedes Ziel $a_{succ\,g_i}$ bzw. $a_{conf\,g_i}$).

Die globale Aktivierungsnormierung wurde bereits von Goetz [1997] durch eine lokale Aktivierungsabzinsung mittels eines Abzinsfaktors β ersetzt. Der Nachteil des Verfahrens ist, daß die Gesamtaktivierung im Netzwerk nicht mehr kontrolliert werden kann, die in MASM-Netzwerken als Anhaltspunkt für die Einstellung der Aktivierungsschwelle verwendet wurde. Der Vorteil ist, daß das Verfahren lokal in den Kompetenzmodulen berechnet werden kann, ohne Information über den Aktivierungszustand aller anderen Kompetenzmodule zu benötigen.

Das Verfahren zur Selektion parallel ausführbarer Verhalten mittels Ressourceknoten hebt gleichzeitig die Beschränkung der MASM-Verhaltensauswahl auf, nicht lokal berechenbar zu sein. Da jedes Kompetenzmodul mit den Ressourceknoten der von ihm verwendeten Ressourcen verbunden ist, kann es lokal entscheiden, ob die benötigten Ressourcen vorhanden sind, oder ob gerade andere Verhalten die Ressource sperren (siehe Kapitel 3.3.1). Des weiteren kennt jedes Kompetenzmodul mittels dieser Verbindungen auch die Aktivierungsschwelle jeder Ressource. Dadurch kann ein Kompetenzmodul lokal ermitteln, ob es genügend Aktivierung besitzt, um ausgeführt zu werden (siehe Kapitel 3.3.2). Kenntnis über die Aktivierung anderer Kompetenzmodule ist für die Verhaltensauswahl nicht mehr nötig.

3.4 Beispiel

Der Aufbau und die Verbindungen eines Verhaltensnetzwerks sind beispielhaft in Abb. 3.3 dargestellt. Das Netzwerk besteht aus zwei Zielen, drei Kompetenzmodulen, einem Ressourceknoten und zwei Wahrnehmungsknoten (Propositionen). Für ein vollständiges Verhaltensnetzwerk fehlen fünf Wahrnehmungsknoten: 'seeBall', 'inOtherHalf', 'staminaLow', 'goal' und 'highStamina'. Weiterhin ist zur besseren Übersicht für jede Art der Verbindung nur ein Beispiel angegeben. Tatsächlich sind im vollständigen Netzwerk insgesamt 17 Verbindungen vorhanden.

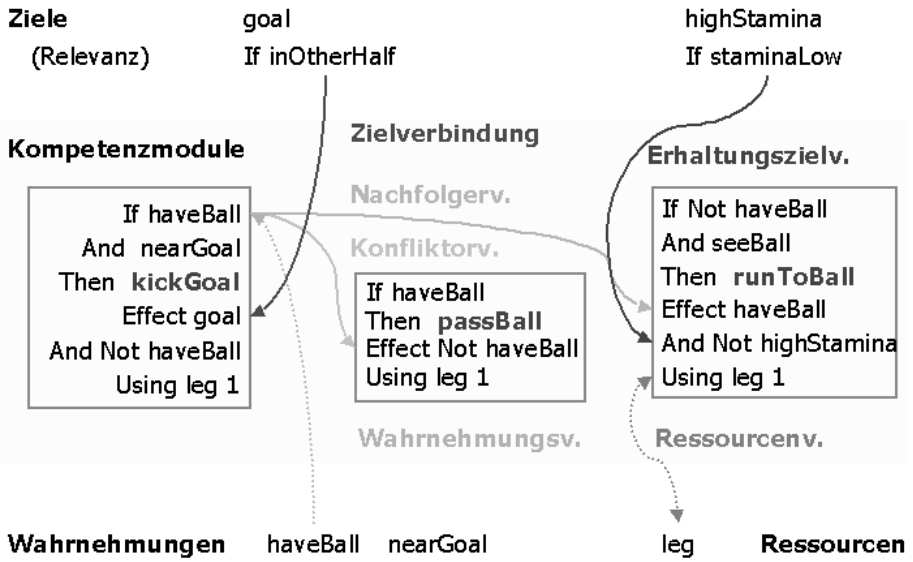

Abbildung 3.3: Beispiel für den Einsatz von Verbindungen zwischen den Knoten eines EBN. Für jede Art von Verbindung ist nur ein Beispiel dargestellt. Des weiteren sind nicht alle notwendigen Wahrnehmungspropositionen abgebildet.

Im folgenden werden die unterschiedlichen Verbindungen zwischen Knoten näher erläutert:

- Jede Proposition einer Vorbedingung, sowie jede Relevanzproposition ist mit der entsprechenden Wahrnehmung verbunden (τ_P-Funktion), wie im Beispiel von 'haveBall' dargestellt.
- 'kickGoal' erhält Aktivierung vom Ziel 'goal', da es einen Effekt 'goal' hat, der das Ziel wahr machen kann.
- Ist die Vorbedingung 'haveBall' vom Kompetenzmodul 'kickGoal' nicht erfüllt, gibt es Aktivierung weiter an das Kompetenzmodul 'runToBall', das die nicht erfüllte Vorbedingung wahr machen kann. 'haveBall' wird so zu einem Unterziel des Netzwerks.
- Der Spieler versucht den bestehenden Zustand 'highStamina' aufrecht zu erhalten, indem dieses Erhaltungsziel das Kompetenzmodul 'runToBall' hemmt, da es Ausdauer (stamina) verbraucht.
- Das Kompetenzmodul 'kickGoal' hemmt 'passBall', falls die Vorbedingung 'haveBall' erfüllt ist, da 'passBall' diese bereits erfüllte Vorbedingung wieder zerstören würde.
- Jedes Verhalten ist weiterhin mit den Ressourceknoten der von ihm verwendeten Ressourcen verbunden. Die Verbindung ist bidirektional, da sowohl Information vom Ressourceknoten zum Verhalten (τ_R-Funktion) als auch Information vom Verhalten zum Ressourceknoten (τ_U-Funktion) fließt.

Kapitel 4

Domänen

Im folgenden werden zwei Umgebungen beschrieben, die hinsichtlich der in Kapitel 1.1.1 genannten Charakteristika gegenüberliegende Endpunkte der Schwierigkeitsskala repräsentieren (Abb. 4.1). Während es sich bei der Blockwelt Domäne um eine deterministische, statische, diskrete und zugängliche Umgebung handelt, ist die RoboCup Domäne nicht-deterministisch, dynamisch, kontinuierlich und nur teilweise zugänglich. Beide Umgebungen wurden für die empirische Untersuchung von Verhaltensnetzwerken verwendet.

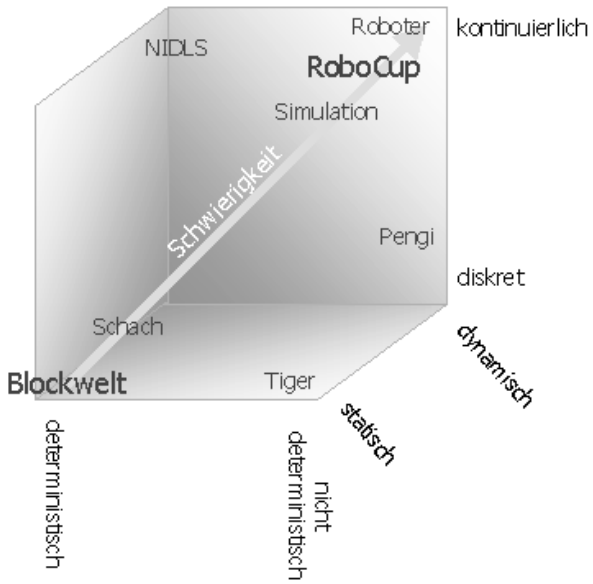

Abbildung 4.1: Kategorisierung der beiden verwendeten Domänen bezüglich der in Kapitel 1.1.1 vorgestellten Kriterien.

4.1 Die klassische Blockwelt Domäne

Die Blockwelt Domäne wurde Ende der 60er Jahre am MIT in Boston eingeführt und gilt als die klassische Mikrowelt für Handlungsplanungsprobleme. Sie wurde

aber auch für verschiedene andere Bereiche der KI wie z.B. maschinelles Lernen [Winston, 1970], Bildverarbeitung [Huffman, 1971] oder Sprachverstehen [Winograd, 1972] eingesetzt.

Objekte der Blockwelt sind eine beliebige aber bestimmte Anzahl von unterscheidbaren Blöcken, ein Tisch, auf dem alle Blöcke liegen können und ein Greifer, der jeweils einen der Blöcke anheben und absetzen kann (Abb. 4.2). Blöcke können zu (beliebig hohen) Türmen aufeinandergesetzt werden, wobei jeweils genau ein Block auf einem anderen Platz findet, d.h. ein Block kann auch nicht auf zwei anderen liegen. Freiheitsgrade des Agenten sind das Hochheben und Absetzen eines Blocks[1].

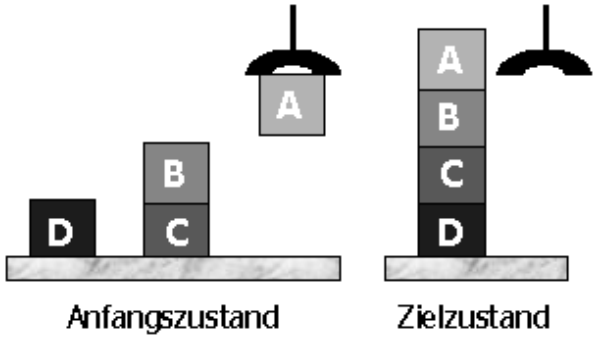

Abbildung 4.2: Beispiel für zwei Zustände in der Blockwelt.

Aktionen sind deterministisch, führen also immer zum gewünschten Ergebnis; ein Block kann nicht aus dem Greifer rutschen oder beim Absetzen vom Turm fallen. Die Domäne ist weiterhin statisch, ein Turm wird nicht durch äußere Einflüsse umstürtzen. Dadurch ist die Handlungskontrolle keinen zeitlichen Einschränkungen unterworfen. Es spielt also keine Rolle, wie lange der Agent für die Entscheidung benötigt. Die Blockwelt ist diskret; ein Block befindet sich entweder auf dem Tisch, im Greifer oder auf einem anderen Block aber niemals irgendwo dazwischen. Dadurch ist die Zahl der Situationen im Vergleich zu kontinuierlichen Umgebungen geringer. Zudem treten die Effekte von Aktionen direkt nach der Ausführung ein. Schließlich ist die Domäne zugänglich, der Agent kann den gesamten Zustand der Umgebung zu jedem Zeitpunkt wahrnehmen, ein Block kann beispielsweise nicht von einem anderen verdeckt sein. Determinismus, Statik und Zugänglichkeit der Umgebung machen es möglich und sinnvoll, weit in die Zukunft zu planen. Der Agent kann den genauen Zustand der Umgebung beliebig weit in der Zukunft vorausplanen. Forderung 2 - 6 aus Kapitel 1.1.3 kommen in der Blockwelt Domäne nicht zum Tragen.

Bei der Implementierung der Verhaltensnetzwerke für diese Domäne mußten einige Besonderheiten berücksichtigt werden. Zum einen können die Verhaltensregeln in der Blockwelt Domäne Variablen enthalten. Diese Variablen stehen für

[1] Es wird gelegentlich zwischen Hochheben, bzw. Absetzen eines Blocks vom Tisch versus von einem anderen Block unterschieden, was zwar die Anzahl der möglichen Aktionen verdoppelt, aber keine echte Erweiterung der Freiheitsgrade bewirkt.

beliebige Blöcke der Domäne. Daher ist es notwendig, die Menge der Instanzen (Blöcke) angeben zu können. Bei der Umsetzung der Netzwerkspezifikation in ein Verhaltensnetzwerk wird dann eine vollständige Instanziierung der Variablen durchgeführt. Jede Verhaltensregel ist also in jeder Kombination von Variableninstanziierungen als Kompetenzmodul im Netzwerk vorhanden.

Zum anderen muß auch der Initialzustand der Umgebung spezifizierbar sein. Dazu kann eine Liste mit den anfänglich wahren Propositionen angegeben werden. Der Zielzustand wird mittels einer Liste von am Ende erwünschten und unerwünschten Propositionen angegeben (siehe Anhang B.3).

4.2 Fußballsimulation RoboCup

Der RoboCup `soccerserver` [Noda, 1995] stellt eine Multiagenten-Umgebung für simulierten, zweidimensionalen Fußball zur Verfügung. Auf einem Spielfeld spielen zwei Mannschaften bestehend aus 10 Feldspielern, einem Torwart und optional einem Trainer Fußball. Ein Spiel ist in zwei Halbzeiten unterteilt, die jeweils 3000 Aktionszyklen (ca. 5 Min.) dauern.

4.2.1 Funktionsweise

Dem Server kommen mehrere Aufgaben zu. Zum einen repräsentiert er die physikalischen Eigenschaften der Objekte in der Domäne. Objekte sind die Spieler, die Trainer, der Ball, die Tore, Landmarken und Linien. Eigenschaften sind unter anderem die Geschwindigkeit, die Position oder die Trägheit von Objekten.

Weiterhin bildet der Server die Schnittstelle zu den Agenten. Er versorgt die Agenten mit lokalen Wahrnehmungen anhand des Zustands der Domäne (Kapitel 4.3.1) und führt Aktionen von Agenten aus, indem der Zustand entsprechend verändert wird. Dabei sind Wahrnehmung und Handlung zyklisch, aber asynchron und werden durch zufälliges Rauschen gestört. Alle 100ms verarbeitet der Server eine Aktion jedes Agenten, alle 150ms sendet er an jeden Spieler eine Wahrnehmung. Aktionen und Wahrnehmungen werden über Netzwerk (UDP/IP[2]) gesendet, wodurch die Agenten auf verschiedenen Rechnern laufen können (Abb. 4.3).

Auf ähnliche Weise liefert der Server die Wahrnehmungen des Trainers. Dieser hat jedoch im Gegensatz zu Spielern globale Weltsicht. Läuft der Server im Trainer-Modus, kann der Trainer die Spieler und den Ball beliebig positionieren, läuft er im normalen Modus, kann der Trainer die Spieler über Zurufe beeinflussen.

Schließlich ist es auch Aufgabe des `soccerservers`, einfache Schiedsrichteraufgaben zu übernehmen. So entscheidet der Server auf Tor, Abseits, Eckball, Abschlag, Einwurf und Spielende.

Zur grafischen Veranschaulichung der Domäne dient der `soccermonitor`, ein Programm, mit dessen Hilfe die aktuelle Spielsituation dargestellt werden kann.

[2]Das UDP/IP Protokoll garantiert im Unterschied zum TCP/IP Protokoll nicht, daß alle Pakete am Zielknoten ankommen. Das deswegen unzuverlässige UDP/IP wurde bewußt gewählt, um der Unzulänglichkeit echter Sensoren Rechnung zu tragen.

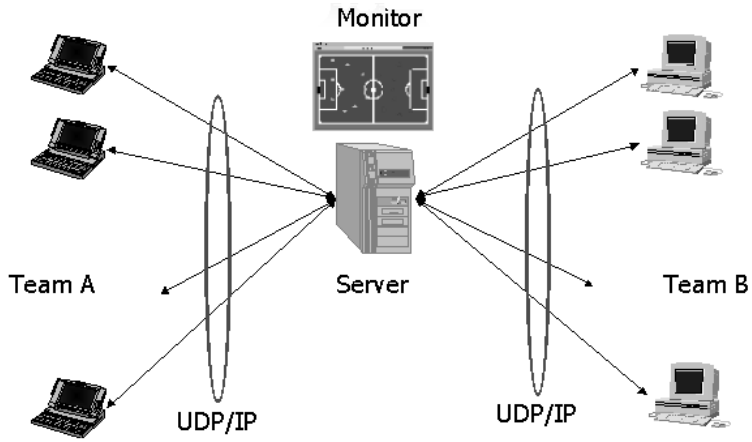

Abbildung 4.3: Schematische Darstellung der RoboCup-Umgebung

Er bietet außerdem die Schnittstelle, die es einem menschlichen Schiedsrichter erlaubt, in das Spielgeschehen einzugreifen. So kann dieser auf Schiedsrichter-Ball entscheiden, Fouls mit einem Freistoß ahnden und bei wiederholtem Foulspiel Spieler des Feldes verweisen. Fouls sind unter anderem, wenn mehrere Spieler den Ball umringen und so dem Gegner den Zugang zum Ball verwehren oder wenn Spieler eine Mauer auf der eigenen Torlinie bilden und somit ein Tor unmöglich machen[3]. Da der `soccermonitor` seine Information ebenfalls über Netzwerk erhält, ist das Programm rechnerunabhängig, kann also sowohl auf einem anderen Rechner als auch unter einem anderen Betriebssystem laufen.

Der `soccerserver` wird inzwischen von vielen Personen und Institutionen als Domäne für die empirische Untersuchung von KI-Methoden verwendet. Er eignet sich insbesondere dazu, verschiedene Ansätze zur Handlungskontrolle direkt miteinander zu vergleichen. 1997 wurde er als Server für die Simulator Liga der ersten Computer-Fußballweltmeisterschaft (RoboCup) in Nagoya/Japan verwendet. Seither wird er ständig erweitert und verbessert und kam in mehreren nationalen und internatinalen Turnieren zum Einsatz. An der RoboCup WM 1999 nahm auch eine Mannschaft der im Rahmen dieser Arbeit erstellten Agenten Teil. Das MagmaFreiburg team (für **M**otivation **A**ction control and **G**oal**m**anagement in **A**gents) [Dorer, 1999b] konnte dabei den 2. Platz erreichen.

4.2.2 Kognitive Adäquatheit

Die RoboCup-Umgebung ist in manchen Punkten einfacher als eine reale Umgebung. Sie ist zunächst zweidimensional, der Ball und die Spieler können sich nur auf einer Ebene bewegen. Weiterhin ist die Zahl der Freiheitsgrade eines Agenten mit sieben (parametrisierbaren) Aktionen deutlich geringer als die eines menschlichen Fußballspielers. Ebenso beschränken sich die Wahrnehmungen eines Agenten auf wenige Eigenschaften des Spielfelds, anderer Spieler und sich

[3]Dies ist möglich, weil es sich um eine zweidimensionale Simulation handelt.

selbst. Schließlich sind auch einige physische Eigenschaften unrealistisch. So kann sich ein Spieler in 100ms um 180° drehen und in manchen Situationen auf eine beliebige Stelle im Spielfeld springen.

Dennoch ist die RoboCup-Umgebung in vielerlei Hinsicht kognitiv anspruchsvoll und realem Fußball ähnlich. Die Umgebung ist dynamisch, d.h. die Situation ändert sich ständig auch ohne das Zutun des Agenten. Dadurch stehen die Entscheidungen eines Agenten unter Zeitdruck. Dieser wird verschärft durch den schnellen Aktionszyklus des Servers von 100ms innerhalb derer sich die Situation ändern kann. Neben der Zeit ist auch die Ausdauer der Spieler eine beschränkte Ressource mit der der Agent haushalten muß. Auch die Schußkraft und die maximale Laufgeschwindigkeit sind begrenzt und in etwa echtem Fußball nachempfunden.

Die Umgebung ist nicht-deterministisch. Die Aktionen des Agenten werden etwas ungenau oder schlimmstenfalls gar nicht ausgeführt. Der Agent kann sich also nicht darauf verlassen, daß die von ihm erwarteten Effekte eines Verhaltens eintreten. Wie beim Fußball sind z.B. Stärke und Richtung eines Schußes nicht perfekt kontrollierbar. Weiterhin ist die RoboCup-Umgebung nicht vollständig zugänglich, d.h. die Verhaltensauswahl basiert auf ungenauen und unvollständigen Wahrnehmungen. Der Agent kann z.B. nur Objekte in Blickrichtung wahrnehmen. Diese Wahrnehmungen sind zudem mit zunehmender Entfernung der wahrgenommenen Objekte ungenau, bzw. entfallen bei weit entfernten Objekten ganz. So kann man z.B. weit entfernte Spieler nicht hören. Schließlich handelt es sich um eine Multiagenten-Umgebung, bei der das Verhalten des Agenten nicht nur den eigenen Nutzen, sondern auch den seiner Mannschaft optimieren soll. Gegnerische Agenten versuchen das zu verhindern.

4.3 Agenten in der Fußballumgebung

Im RoboCup werden drei Klassen von Agenten unterschieden: Feldspieler, Torwart und Trainer. Wie bereits erwähnt, besitzt der Trainer globale Weltsicht, kann aber bei Spielen nicht physisch, sondern nur verbal in das Spielgeschehen eingreifen. Da in den Untersuchungen zu Verhaltensnetzwerken kein Trainer eingesetzt wurde, wird hier nicht näher darauf eingegangen.

Feldspieler und Torwart unterscheiden sich lediglich im größeren Freiheitsgrad des Torwarts, der im eigenen Strafraum den Ball festhalten kann. Ansonsten sind beide im wesentlichen durch ihre Position auf dem Spielfeld, ihre Geschwindigkeit, durch Blickrichtung und Modus (siehe Kapitel 4.3.1) und durch ihre Ausdauer beschrieben. Letztere vermindert sich beim Laufen und regeneriert sich langsam, wenn der Agent stehen bleibt. Je öfter sich der Agent stark verausgabt, desto langsamer regeneriert er sich.

In Abbildung 4.4 ist der Aufbau der Verhaltenskontrolle eines von uns verwendeten Agenten dargestellt. Zentraler Bestandteil ist das für die Verhaltensauswahl zuständige erweiterte Verhaltensnetzwerk. Der Wahrheitswert jedes Wahrnehmungsknoten des Netzwerks wird aus der Information des globalen Weltmodells berechnet. Dieses wird sowohl aus der vom Server an den Agenten gelieferten Wahrnehmung generiert, als auch durch Antizipation der Effekte

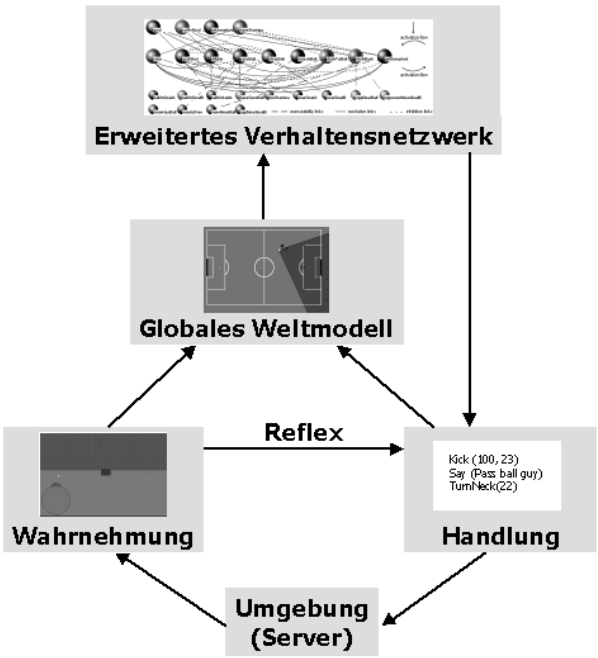

Abbildung 4.4: Modell der MagmaFreiburg Agenten.

des eigenen Handelns aktualisiert. Das Modell erlaubt auch Reflexe, also direkt auf Wahrnehmungen folgende, nicht unterdrückbare Handlungen. Im folgenden sind die Aufgaben und die Funktionsweise der einzelnen Module im Bezug auf die Fußballumgebung genauer beschrieben.

4.3.1 Wahrnehmung

Der Agent erhält lokale, d.h. unter Berücksichtigung von Position und Blickrichtung des Agenten gewonnene Wahrnehmung in Form einer Zeichenkette vom Server. Die Wahrnehmung verschiedener Spieler wird sich also immer unterscheiden. Es werden drei Arten von Wahrnehmung unterschieden: 'visuelle', 'auditive' und 'propriozeptive' Wahrnehmung.

Der Agent erhält seine 'visuelle' Wahrnehmung in drei unterschiedlichen Modi, die er selbst aktiv wählen kann. Im fokusierten Modus erhält er alle 75 ms eine Wahrnehmung, der sichtbare Bereich ist dabei aber auf 45° (Blickrichtung $\pm 22.5°$) beschränkt. Im normalen Modus beträgt der sichtbare Bereich 90°, der Agent empfängt aber nur alle 150 ms eine Wahrnehmung. Der Weitwinkel-Modus schließlich liefert alle 300 ms 180° Sicht. Der Agent kann also seine visuelle Wahrnehmung mehr oder weniger fokusieren. Diese besteht aus der Art des sichtbaren Objekts sowie aus der Entfernung und Richtung. Entfernung und Richtung können um so ungenauer sein, je weiter entfernt sich ein Objekt befindet. Zusätzlich entfällt bei entfernten Spielern die Spielernummer, bei weit entfernten Spielern auch die Information über die Mannschaftszugehörigkeit des Spielers.

(see 73 ((goal r) 66 -24) ((flag c) 28.2 25 -0.564 0.4) ((flag c b) 61.6 34) ((flag r b) 85.6 -3) ((flag p r t) 44.3 -39) ((flag p r c) 51.4 -16) ((flag p r b) 64.1 -1) ((flag g r t) 64.7 -27) ((flag g r b) 67.4 -21) ((flag b l 10) 66 43) ((flag b 0) 66.7 34) ((flag b r 10) 68.7 26) ((flag b r 20) 71.5 18) ((flag b r 30) 75.9 11) ((flag b r 40) 81.5 5) ((flag b r 50) 87.4 0) ((flag r t 20) 65.4 -42) ((flag r t 10) 67.4 -33) ((flag r 0) 70.1 -25) ((flag r b 10) 75.2 -18) ((flag r b 20) 80.6 -12) ((flag r b 30) 86.5 -7) ((Player) 2.5 -124) ((player TeamDorer 1) 12.2 18 -0.488 0.5 -143) ((player TeamMaes 2) 36.6 25 -0.732 0.7 149) ((line b) 81.5 -48)

Abbildung 4.5: 'Visuelle' Wahrnehmung eines Agenten vom Server

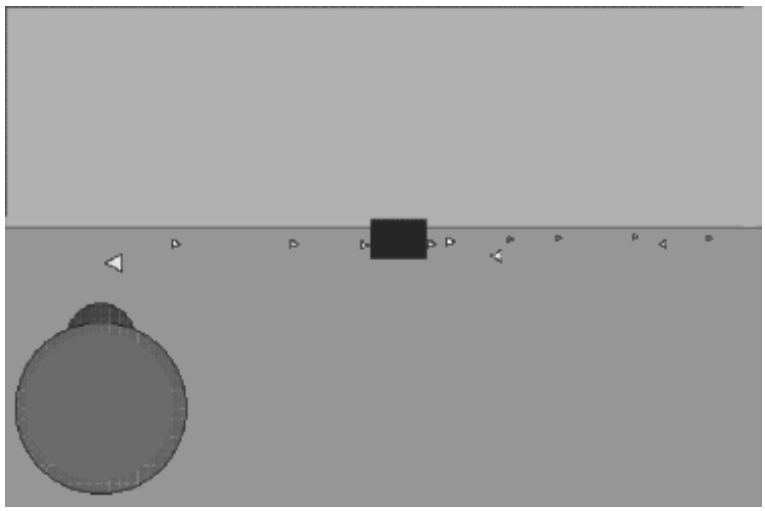

Abbildung 4.6: Grafische Veranschaulichung der relativen Wahrnehmung. Rechteck = Tor, heller Kreis = Mitspieler, dunkler Kreis = Ball, Dreiecke = Landmarken, Linie = Horizont.

Weiterhin 'hört' der Agent Zurufe anderer Spieler oder des Trainers, die sich innerhalb eines begrenzten Radius um den Agenten befinden. Die Entscheidungen des Schiedsrichters kann der Agent jedoch überall 'hören'.

Als 'propriozeptiv' kann schließlich die Wahrnehmung der momentan verbleibenden Ausdauer des Agenten sowie der Betrag der Eigenbewegung bezeichnet werden.

Neben dem Parsen der Wahrnehmung vom Server ist die wichtigste Aufgabe des Wahrnehmungsmoduls die Selbstlokalisation. Der Agent erhält keine Information über seine globale Position auf dem Spielfeld, sondern nur relative Angaben zu sichtbaren Objekten. Verschiedene Aufgaben vereinfachen sich aber, wenn der Agent seinen genauen Standpunkt kennt. So ist es beispielsweise schwierig, ohne globale Koordinaten eine Abseitsposition zu erkennen, da die Abseitslinie durch eine quer zum Spielfeld durch den letzten Feldspieler

der abwehrenden Mannschaft verlaufende Linie festgelegt ist. Weiterhin kann ein Agent ohne globale Lokalisation leichter die Orientierung verlieren, insbesondere, wenn in seinem sichtbaren Bereich nur wenige Landmarken zu sehen sind.

Seine globale Position kann der Agent mit Hilfe der sichtbaren Landmarken berechnen, deren absolute Position er kennt. Mit Hilfe von Triangulierung ist so eine ungefähre Positionsbestimmung möglich. Ungefähr deshalb, weil die Angaben über Richtung und Entfernung der Landmarken ungenau sind. Zur Erhöhung der Genauigkeit kann dann noch die bisherige Position des Agenten sowie die durch die Geschwindigkeit erwartete neue Position beitragen.

4.3.2 Reflexe

Direkt nach der Verarbeitung der Wahrnehmungen vom Server wird überprüft, ob die Bedingungen für einen Reflex erfüllt sind. Reflexe sind nicht unterdrückbare Handlungen, die direkt durch einen Reiz ausgelöst werden können. In der Fußballumgebung wurde zum Beispiel die Möglichkeit, einen Spieler direkt nach einem Tor an eine beliebige Stelle in der eigenen Hälfte zu platzieren, als Reflex implementiert, da dieses Verhalten immer zu diesem spezifischen Reiz ausgeführt werden soll. Wurde ein Reflex ausgelöst, findet anschließend keine Verhaltensauswahl durch das Verhaltensnetzwerk statt. Dadurch wird der Tatsache Rechnung getragen, daß sich Reflexe gegenüber höheren kognitiven Handlungen durchsetzen.

4.3.3 Globales Weltmodell

Kennt der Agent seine globale Position auf dem Spielfeld, kann er auch alle anderen sichtbaren Spieler in sein globales Weltmodell eintragen, das bereits alle statischen Objekte, wie die Tore, enthält. Dadurch kann der Agent auch Objekte berücksichtigen, die nicht sichtbar sind und so beispielsweise einen Torschuß wagen, ohne das Tor sehen zu müssen. Nicht sichtbare, bewegliche Objekte wie etwa andere Spieler oder der Ball können in dieser Karte nur für eine gewisse Zeit[4] eingetragen bleiben, da sich deren Position schnell ändern kann. Ein Objekt wird auch dann aus dem globalen Weltmodell gelöscht, wenn es an einer Stelle vermutet wird, die der Agent (inzwischen) einsehen kann, es sich aber nicht dort befindet.

Weiterhin verbessern die Spieler untereinander ihr Weltmodell, indem sie miteinander kommunizieren und so die Position der von ihnen wahrgenommenen Objekte anderen mitteilen. Da die Bandbreite der Kommunikation stark beschränkt ist - ein Spieler kann nur alle 2 Sekunden etwas (512 Byte) sagen - ist diese Information jedoch meist älter und damit ungenauer als die der eigenen Wahrnehmung. Ein Spieler kann dadurch aber auch andere Spieler oder den Ball wahrnehmen, wenn sie sich hinter ihm befinden.

Mit Hilfe früherer Ballpositionen wird die Geschwindigkeit des Balls und der Spieler bestimmt. Das Weltmodell kann damit auch um die erwarteten zukünfti-

[4]Bei den hier verwendeten Agenten betrug diese Zeit 3 Sekunden in denen das Zuverlässigkeitsmaß für die Position linear sank

gen Ballpositionen erweitert werden. Das ist zum Beispiel beim Abfangen eines Balls hilfreich, wo der Agent nicht zur aktuellen Ballposition, sondern zu einer zukünftigen möglichst nahen Ballposition rennen muß, um schnell am Ball zu sein.

Schließlich wird das globale Weltmodell auch durch Antizipation der Effekte des eigenen Handelns verbessert. Schießt der Agent den Ball, dann werden die erwarteten zukünftigen Ballpositionen entsprechend der Schußrichtung und Stärke berechnet. Das ist notwendig, da der Wahrnehmungszyklus des Servers 150ms beträgt, während der Handlungszyklus nur 100ms lang ist. Der Agent muß also immer wieder Entscheidungen treffen, für die noch keine neuen Wahrnehmungen vorhanden sind.

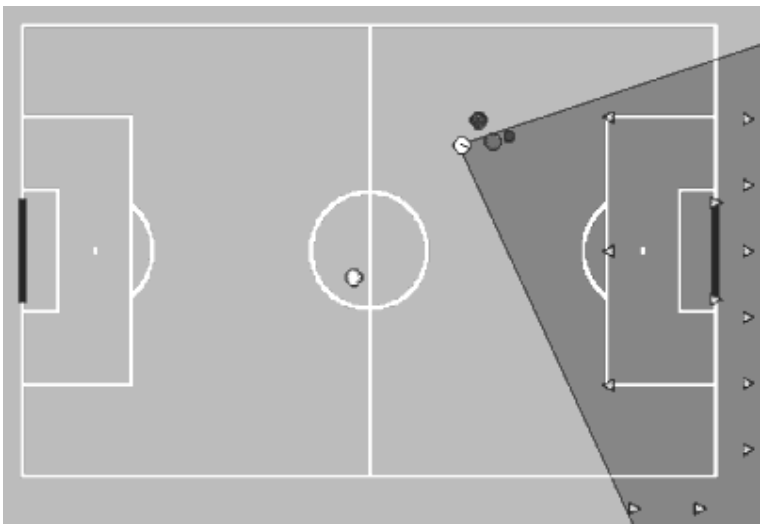

Abbildung 4.7: Grafische Veranschaulichung der globalen Karte. Dunkler Bereich = sichtbarer Bereich des Agenten, weisser Kreis = Agent selbst, heller Kreis = Mitspieler, heller Kreis im sichtbaren Bereich= Gegenspieler, schwarzer Kreis = Gegenspieler, der bereits längere Zeit nicht gesehen wurde, aber dort vermutet wird, dunkler Kreis im sichtbaren Bereich = Ball, Dreiecke = Landmarken.

Teil des globalen Weltmodells sind auch die Index-Objekte (indexical functional objects [Agre und Chapman, 1987]). Anstatt die Spieler absolut zu benennen, werden Spieler mit Hilfe ihrer Funktion in der momentanen Situation indiziert. Ein Spieler wird also beispielsweise nicht 'Mitspieler9', sondern 'MitspielerAmBall' genannt, wenn er den Ball besitzt, oder 'FreierMitspieler', wenn er der am besten anspielbare Mitspieler ist. Die Zahl der Verhaltensregeln kann dadurch auch ohne die Verwendung von Variablen stark eingeschränkt werden. Man benötigt z.B. nicht für jeden Mitspieler ein eigenes Abspielverhalten, sondern nur ein Abspielverhalten, das dem 'FreienMitspieler' den Ball zuspielt.

Das globale Weltmodell wird dann für die Berechnung der Wahrnehmungspropositionen des Verhaltensnetzwerks verwendet. Die Speicherung der bisherigen Wahrnehmung in einem globalen Weltmodell erleichtert dabei die Berech-

nung einiger Propositionen, bzw. macht diese erst möglich. So kann der Agent selten anhand nur einer (der aktuellen) Wahrnehumng entscheiden, ob er sich in einer Abseitsposition befindet, da er nicht den gesamten relevanten Bereich mit einem Blick überblicken kann. Die Wahrnehmungspropositionen des Verhaltensnetzwerks repräsentieren also zum Teil relativ abstrakte Zustände der Umgebung, für die eine entsprechende Vorverarbeitung der Basiswahrnehmungen des Agenten notwendig ist.

4.3.4 Verhaltensnetzwerk

In Abbildung 4.8 ist ein Beispiel für ein Verhaltensnetzwerk dargestellt, wie es für einige Spiele mit zwei Spielern je Mannschaft verwendet wurde. In den meisten Untersuchungen im Rahmen dieser Arbeit wurde die Zahl der Spieler auf zwei beschränkt, um zum einen kein Rollenverhalten der Agenten einführen zu müssen und somit identische Verhaltensnetzwerke für alle Spieler verwenden zu können und zum anderen, um eine ausreichende Performance des Client-Rechners zu gewährleisten. Die maximale Ausdauer dieser Spieler wurde daher erhöht, Abseits wurde nicht gepfiffen.

Die Ziele eines Agenten bei diesem Netzwerk sind (Zielbedingung, Wichtigkeit, Relevanzbedingung):

- Tore erzielen, 0.9, Ball nahe beim gegnerischen Tor (wahr, wenn sich der Ball näher als 15 m beim gegnerischen Tor befindet, falsch, wenn er sich mehr als 70 m vom gegnerischen Tor befindet mit linearer Interpolation dazwischen)

- Gegentore vermeiden, 0.8, Ball nahe beim eigenen Tor (entsprechend wie oben für das eigene Tor; die Relevanzbereiche überlappen sich also)

- Ausdauer besitzen, 0.6, habe keine Ausdauer (wahr bei weniger als 200 von maximal 4000 Ausdauerpunkten, falsch bei mehr als 2200)

- Anspielbar sein, 0.5, Mitspieler hat den Ball (wahr, wenn der Mitspieler der am nächsten zum Ball stehende Spieler und weniger als 1 m vom Ball entfernt ist; falsch, wenn ein anderer Spieler näher zum Ball steht, oder der Mitspieler mehr als 9 m vom Ball entfernt ist)

Weiterhin besitzt der Agent neun Verhalten: 'sich ausruhen', 'Torschuß', 'mit dem Ball dribbeln', 'den Mitspieler anspielen', 'Befreiungsschlag', 'den Ball suchen', 'zum Ball rennen', 'zum eigenen Tor rennen' und 'sich freilaufen'. Dabei sind es die beiden Verhalten 'sich freilaufen' und 'den Mitspieler anspielen', die ohne explizite Kommunikation für emergentes Mannschaftsverhalten sorgen.

Schließlich bestand die Wahrnehmung des Agenten aus 13 Wahrnehmungspropositionen: 'Ball ist nahe dem gegnerischen Tor', 'Ball ist nahe dem eigenen Tor', 'Ball ist im Einflußbereich, 'habe Ball (gerade) gesehen', 'habe wenig Ausdauer', 'bin nahe am gegnerischen Tor', 'bin nahe am eigenen Tor', 'Gegner im Ballbesitz', 'Gegner näher bei unserem Tor', 'Mitspieler im Ballbesitz', 'Mitspieler steht frei', 'Mitspieler steht näher zum Ball', 'Mitspieler steht näher zum gegnerischen Tor'.

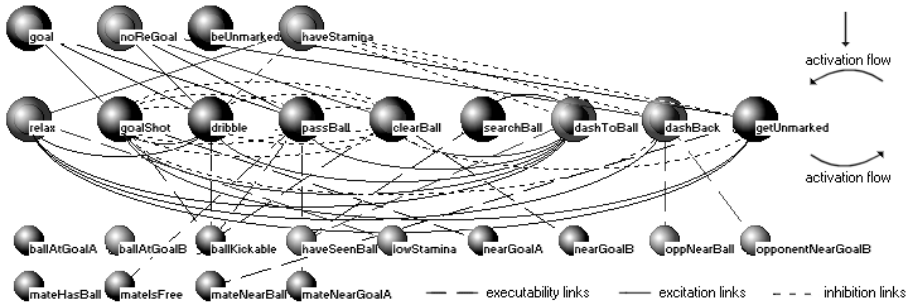

Abbildung 4.8: Beispiel eines Verhaltensnetzwerks für Spiele mit vier Spielern

Der Wahrheitswert dieser Propositionen wird bei jedem Eintreffen von Wahrnehmungen aus diesen neu berechnet. Bei jedem Aktionstakt wird dann solange Aktivierung ausgetauscht, bis Verhalten selektiert wurden. Diese Verhalten werden vom Handlungsmodul ausgeführt.

4.3.5 Handlung

Der Agent kann komplexe Verhalten in mehrere Aktionen umsetzen, die dann als Aktionsfolge gespeichert werden. Ist eine solche Aktionsfolge noch nicht vollständig abgearbeitet, wird die nächste darin enthaltene Aktion zum Server gesendet und keine neue Verhaltensauswahl getroffen. Aktionsfolgen haben den Vorteil, daß sie geringeren Rechenaufwand benötigen und im Fall von nicht vorgegebenem Aktionstakt schneller ausgeführt werden können. Der Nachteil ist, daß Änderungen in der Situation während der Ausführung von Aktionsfolgen nicht berücksichtigt werden. Daher sollte die Länge solcher Aktionsfolgen auf wenige Aktionen beschränkt sein. Im Rahmen des RoboCup wurden lediglich für das 'hardKick' Verhalten solche Aktionsfolgen benutzt. Um den Ball besonders stark zu schießen, wird in bestimmten Situationen eine Aktionsfolge von zwei direkt aufeinanderfolgenden Schüssen generiert, die dann unbedingt ausgeführt werden.

Ist keine Aktion einer Aktionsfolge auszuführen, erfolgt die Verhaltensauswahl mit Hilfe des Verhaltensnetzwerks. Die oben beschriebenen Verhalten des Agenten sind zum Teil recht komplexe Verhaltensweisen, die erst noch auf die einfacheren Aktionen des Agenten abgebildet werden müssen. Dazu können komplexe Verhalten aus weniger komplexen Verhalten aufgebaut sein. Das komplexe Verhalten Torschuß basiert beispielsweise auf dem einfacheren und generelleren Verhalten Schuß, das im im Gegensatz zur Aktion 'kick' Kollisionen von Ball und Agent vermeidet. Diese fortgeschrittenen Verhalten werden schließlich auf eine der Situation angepaßte Aktion abgebildet und zum Server gesendet.

4.4 RoboCup99

Für den RoboCup99 wurde die Anzahl von Verhalten, Wahrnehmungen und Zielen der Verhaltensnetzwerke erhöht [Dorer, 1999b]. Insbesondere wurden wei-

tere Verhalten für Mannschaftsstrategien eingeführt. Beispiele dafür sind die Kommunikation der Spieler untereinander sowie der Aufbau einer Abseitsfalle.

Dazu war es auch nötig, die Zahl der Index-Objekte (indexical functional objects [Agre und Chapman, 1987]) zu erhöhen. Index-Objekte werden verwendet, um Objekte nicht absolut, sondern nach ihrer Funktion in der aktuellen Situation zu referenzieren (siehe Kapitel 4.3.3 sowie 6.4.3). Dadurch kann die Zahl der benötigten Verhaltensregeln deutlich reduziert werden. So wären z.B. bei absoluter Indizierung der Tore zwei Torschußverhalten notwendig: eins für das linke Tor und eins für das rechte Tor. Durch die Einführung des Index-Objekts 'OtherGoal', das je nach Situation das linke oder das rechte Tor indiziert, ist nur ein Torschußverhalten notwendig.

Die verwendeten Index-Objekte waren:

- 'OpponentAtBall', der Gegenspieler, der am nächsten zum Ball steht.
- 'OpponentAtMe', der Gegenspieler, der am nächsten zum Agenten steht.
- 'BackmostOpponent', der Gegenspieler, der am nächsten beim gegnerischen Tor steht (nur x-Koordinate).
- '2BackmostOpponent', der zweithinterste Gegenspieler, also der, der die Abseitslinie festlegt.
- 'FrontmostTeammate', der vorderste Mitspieler, also der, der am ehesten im Abseits steht.
- 'BackmostTeammate', der hinterste Mitspieler.
- '2BackmostTeammate', der zweithinterste Mitspieler, der die eigene Abseitslinie festlegt.
- 'TeammateAtBall', der nächste Mitspieler zum Ball.
- '2TeammateAtBall', der zweitnächste Mitspieler zum Ball.
- 'TeammateAtMe', der nächste Mitspieler beim Agenten.
- 'FreeTeammate', der Mitspieler, der am besten anspielbar ist.
- 'OwnGoal', das zu verteidigende Tor.
- 'OtherGoal', das gegnerische Tor.

Weiterhin wurde beim Spiel mit elf Spielern Rollenverhalten eingeführt. Es gab unterschiedliche Verhaltensnetzwerke für den Torwart, Abwehrspieler, Mittelfeldspieler und Stürmer. Diese unterschieden sich sowohl in der Anzahl und Art von Verhalten, als auch in der Wichtigkeit der Ziele. Die Wichtigkeit des Ziels, Tore zu erzielen, war beim Torwart gering, während sie bei Stürmern maximal war. Die Abwehrspieler besaßen das Verhalten, eine Abseitsfalle aufzubauen, während Stürmer ein Verhalten hatten, sich aus einer Abseitsposition zurückzuziehen. Mittelfeldspieler konnten beide Verhalten ausführen.

Im folgenden sind die einzelnen Rollen der Agenten näher erläutert. Die dazugehörigen Verhaltensnetzwerke sind im Anhang B.2 aufgeführt.

4.4.1 Torwart

Der Torwart unterscheidet sich neben einer abgewandelten Priorisierung der Ziele vor allem durch drei zusätzliche Verhalten gegenüber den anderen Agenten. Zum einen kann der Torwart den Ball fangen und dadurch den Gegenspielern den Zugang zum Ball verwehren. Da der Ball nur alle fünf Zyklen gefangen werden kann, muß die Zuverlässigkeit dieses Verhaltens besonders hoch sein. Greift der Torwart zu früh nach dem Ball, ist dieser also noch zu weit entfernt, kann erst nach einer halben Sekunde ein weiterer Versuch unternommen werden, den Ball zu fangen. Starke Schüße sind dann aber bereits im Tor. Eine hohe Zuverlässigkeit wurde unter anderem dadurch erreicht, daß die τ_P-Funktion für die Proposition 'ballIsCatchable' nicht im gesamten Fangbereich des Torwarts von 2 m, sondern nur bis zu einer Entfernung des Balls von 1.8 m den Wert 1.0 lieferte, und bei weiterer Entfernung linear bis zu einem Wert 0.0 bei zwei Metern sank.

Zum zweiten kann der Torwart nach dem Fangen des Balls eine beliebige Position innerhalb des eigenen Strafraums annehmen. Diese Möglichkeit wurde zum einen dazu genutzt, Ausdauer einzusparen, zum anderen wurde dabei auch das Spiel auf die andere Seite verlagert. Fand der abgefangene Angriff auf der linken Seite statt, wechselte der Torwart anschließend auf die rechte Seite. Dadurch wurden die Spieler eingesetzt, die zuvor weniger gerannt waren.

Zum dritten ist es die Aufgabe des Torwarts, den Winkel zwischen Ball und Tor zu verkürzen, einem Angreifer also möglichst wenig Torfläche offen zu lassen. Dazu ist es notwendig, daß sich der Torwart entsprechend der Ballposition vor dem Tor hin und her bewegt. Gleichzeitig soll er dabei aber nie den Ball aus den Augen verlieren. Das konnte mit der Möglichkeit, den Kopf unabhängig vom Körper zu drehen, erreicht werden. Besonders für den Torwart war also die parallele Ausführung von Bewegungen und Kopfdrehungen wichtig. Konnte auch durch Kopfdrehung der Ball nicht im sichtbaren Bereich gehalten werden, rannte der Torwart als einziger Spieler auch rückwärts, obwohl das die doppelte Menge an Ausdauer verbraucht.

4.4.2 Abwehr

Die Koordination der Spieler untereinander ist implizit und findet dadurch statt, daß die Spieler anhand gemeinsam wahrnehmbarer, vorher festgelegter Situationen bestimmte Rollenverhalten ausführen (lockerroom agreement [Stone *et al.*, 1999]). Im Fall der Abseitsfalle ist festgelegt, daß ein Abwehrspieler genau dann aufrückt, wenn sich der Ball mehr als eine bestimmte Strecke (z.B. 5 m in Längsrichtung zum Spielfeld) vor dem Agenten befindet. Zusätzlich darf sich kein Mitspieler deutlich hinter dem Agenten befinden. Dadurch wird gewährleistet, daß die gesamte Abwehr gemeinsam aufrückt, was das kritischste Kriterium für eine Abseitsfalle darstellt. Das gemeinsame Signal ist die Position des Balls. Kann ein Mitspieler nicht aufrücken, weil er z.B. nicht genügend Ausdauer besitzt, werden auch alle anderen Abwehrspieler in der Defensive verharren.

4.4.3 Mittelfeld

Die Mittelfeldspieler sind eine Kombination aus Abwehrspielern und Stürmern. Sie müssen sowohl in der Lage sein, Abseitspositionen zu vermeiden, als auch in der Defensive eine Abwehrfalle aufzubauen. Sie müssen Tore erzielen können sowie Tore verhindern helfen. Und wie im richtigen Fußball haben die Mittelfeldspieler auch die weitesten Strecken zurückzulegen. Besonders bei diesen ist also eine gute Ressourcenkontrolle wichtig. Dies wird erreicht, indem ein Spieler nicht zum Ball läuft, wenn er nicht ausreichend Ausdauer besitzt. Gleichzeitig kennen die anderen Spieler durch Kommunikation den Ausdauerzustand aller übrigen Spieler und können erkennen, wenn ein Spieler müde ist. Der am nächsten positionierte Spieler übernimmt dann die Rolle des sich ausruhenden Spielers und rennt selbst zum Ball.

4.4.4 Sturm

Ein Stürmer unterscheidet sich von einem Abwehrspieler in drei Punkten. Zum einen muß ein Stürmer vermeiden, in einer Abseitsposition zu stehen. Da ein Spieler in der Regel nicht den gesamten für eine Abseitsposition relevanten Bereich überblicken kann, ist besonders für die Erkennung von Abseitspositionen das globale Weltmodell wichtig. Mit Hilfe der Information im globalen Weltmodell kann eine Abseitsposition leicht festgestellt werden. Da sich die Position der anderen Spieler ständig ändert, muß die Information des Weltmodells aktiv aufgefrischt werden, indem sich der Spieler von Zeit zu Zeit umsieht. Stellt ein Stürmer fest, daß er sich im Abseits befindet, läuft er in Richtung des eigenen Tors.

Weiterhin führt ein Stürmer eher ein Dribbling aus. Die Gefahr durch einen Ballverlust ist in der eigenen Hälfte wesentlich größer als in der gegnerischen Hälfte. Während ein Abwehrspieler den Ball also abspielt oder einen Befreiungsschlag ausführt, wenn er angegriffen wird, versucht ein Stürmer auch mal ein Dribbling, wenn kein Mitspieler anspielbar ist.

Schließlich führen Stürmer bei Abschlägen des Gegners ein abgewandeltes Positionierungsverhalten aus, das der Tatsache Rechnung trägt, daß der Strafraum in dieser Situation nicht betreten werden darf.

4.5 Testumgebung für Spielserien

Da die RoboCup-Domäne nicht-deterministisch ist, konnten Ergebnisse nur statistisch aus Serien von Spielen ermittelt werden. Um solche Serien automatisch ausführen zu können, wurde eine spezielle Umgebung entwickelt, die die Kontrolle aller benötigten Programme übernimmt und statistische Daten zentral archiviert (Abb. 4.9).

Der Controller, ein erweiterter `soccermonitor`, startet über eine Telnet-Verbindung den `soccerserver` auf einer Sun. Danach startet er die Agenten, die entsprechend den aktuellen Versuchsbedingungen über die Kommandozeile parametrisiert werden können. Über eine TCP/IP Verbindung bestätigt der Agent das erfolgreiche Anmelden (UDP/IP) beim `soccerserver`. Über diese

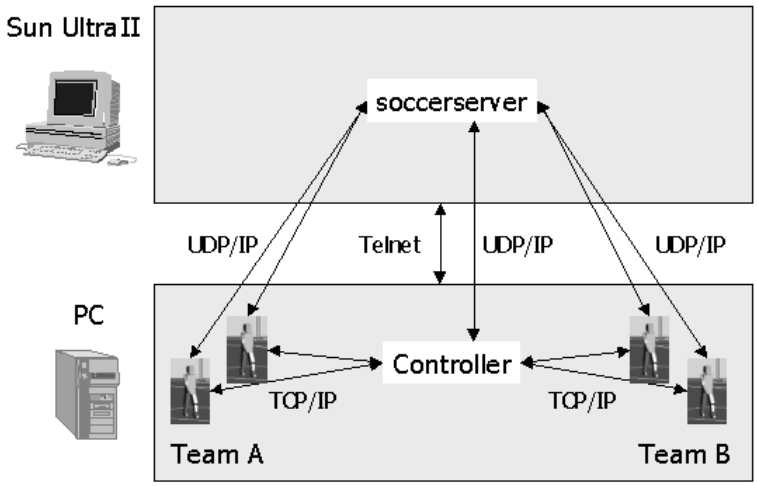

Abbildung 4.9: Testumgebung für Serienspiele in der RoboCup-Umgebung.

Verbindung werden am Ende eines Spiels auch die statistischen Daten jedes Spielers, wie etwa die Anzahl von Ausführungen jedes Verhaltens, sowie die Nachricht zum Beenden des Programms übertragen.
Über die Standard `soccermonitor`-Schnittstelle (UDP/IP) empfängt der Controller Information über den Spielstand und die Position der Spieler und des Balls vom `soccerserver`. Diese wird sowohl für das aktive Eingreifen ins Spiel im Falle von festgefahrenen Spielsituationen als auch für das Aufzeichnen der Spielergebnisse genutzt. Alle Daten einer Serie werden schließlich in einer Datei gespeichert.

Kapitel 5

Empirische Untersuchungen

In diesem Kapitel sind empirische Untersuchungen zur Handlungskontrolle mit erweiterten Verhaltensnetzwerken beschrieben. Sie sollen klären, ob die in Kapitel 3 vorgeschlagenen Erweiterungen den gewünschten Einfluß auf die Handlungskontrolle haben, bzw. eine Verbesserung der Handlungskontrolle bewirken.

Für die empirischen Untersuchungen wurde im wesentlichen die RoboCup-Umgebung [Noda, 1995] herangezogen. Neben der Berücksichtigung mehrerer gleichzeitiger Ziele wie etwa ein Tor zu erzielen, Gegentore zu verhindern, nicht im Abseits zu stehen, muß eine erfolgreiche Handlungskontrolle im Gegensatz zu statischen Umgebungen hier auch schnell und situationsangepaßt sein und auf unvorhergesehene Dinge reagieren können (siehe Kapitel 4.2.2).

Da die RoboCup-Domäne dynamisch und nicht-deterministisch ist, können Ergebnisse nur statistisch aus Serien von Spielen gewonnen werden. Dazu wurden in der Regel 30 simulierte Fußballspiele durchgeführt, wobei zwei Mannschaften mit meist zwei Spielern gegeneinander spielten. Im Unterschied zu Spielen mit elf Spielern wurde kein Abseits verwendet und der Parameter, der die Erholung der Spieler steuert, wurde erhöht, damit die Spieler genügend Ausdauer besaßen, zu viert das ganze Spielfeld zu benutzen. Die Spieler einer Mannschaft waren immer identisch (bis auf die Startposition), die Spieler der beiden Mannschaften unterschieden sich nur in dem zu untersuchenden Parameter, bzw. in der zu untersuchenden Variation (siehe folgende Kapitel).

Die Daten eines Spiels umfaßten die Kennung des Spiels, die Anzahl Tore beider Mannschaften sowie die Anzahl Ausführungen jedes Verhaltens, die mittlere Anzahl ausführbarer Module und die Anzahl Verhaltenswechsel jedes Agenten. Ergebnis einer solchen Serie von Spielen war z.B. ein Vergleich der mittleren Anzahl Tore beider Mannschaften. Als statistische Tests für die Signifikanz von Ergebnissen wurden T-Test mit gepaarten Stichproben auf unterschiedlichen Mittelwert mit $\alpha = 0.01$ verwendet.

5.1 Optimierung der Parameter

Die Parameter in Π zur Steuerung der Aktivierungsausbreitung und der Verhaltensauswahl (Kapitel 3.1) sind domänen-abhängig, müssen also an jede Domäne angepaßt werden. Diese Anpassung erfolgte in separaten Vorversuchen, um

während der eigentlichen Untersuchungen konstante Parametereinstellungen und damit besser kontrollierbare Versuchsbedingungen im Gegensatz zu einer adaptiven Einstellung der Parameter [Maes, 1991] gewährleisten zu können (siehe Kapitel 2.4).

In Serien von Spielen wurde ermittelt, welche Parametereinstellung die besten Ergebnisse liefert. Jeweils ein Parameter wurde bei den Agenten einer Mannschaft innerhalb seines Definitionsbereichs variiert. Für jede Variation wurden 30 Spiele gegen eine Kontrollmannschaft durchgeführt, bei der dieser Parameter einen konstanten Wert hatte, die sonst aber identisch war. Qualitätsmaß war der Torunterschied zur Kontrollmannschaft. Für die Untersuchung des ersten Parameters wurden die Werte der übrigen Parameter auf beliebige Werte gesetzt. Die weiteren Parametervariationen verwendeten dann die zuvor gefundenen optimalen Parameterwerte. Da die Parameter nicht unabhängig voneinander sind, wurde dieser Prozeß solange wiederholt, bis die Differenz von bisherigem und neuem optimalen Parameterwert klein (≤ 0.1) waren. Dies führte zu Schaubildern wie in Abbildung 5.1 dargestellt.

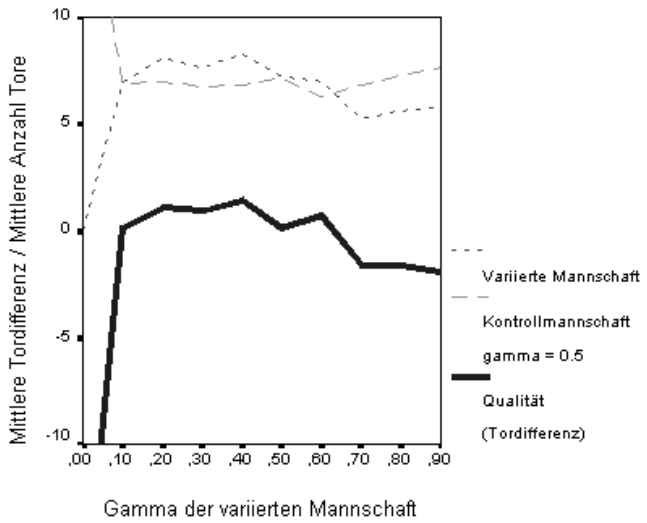

Abbildung 5.1: Qualität einer Mannschaft mit zwei Spielern als Funktion der Aktivierung durch Ziele γ. Die Kontrollmannschaft (static team) war identisch, verwendete aber eine konstante Zielaktivierung von $\gamma = 0.5$.

Bemerkenswert ist zum einen, daß die Qualität der Verhaltenskontrolle ohne Motivation durch Ziele ($\gamma = 0$) schlecht ist, obwohl nur etwa drei Kompetenzmodule im Mittel ausführbar waren, aus denen eins selektiert wurde. Zum anderen ist die mittlere Anzahl Tore über einen großen Parameterbereich hoch, d.h. es ist nicht schwierig, eine funktionierende Parametereinstellung zu finden.

Die so ermittelten optimalen Parametereinstellungen für Spiele mit zwei Spielern sind in Tabelle 5.1 dargestellt. Diese Werte wurden auch für Spiele mit elf Spielern und größeren Verhaltensnetzwerken im RoboCup99 verwendet.

γ	δ	β	θ
0.5	0.4	0.15	0.4

Tabelle 5.1: Verwendete Parameter für EBNs in der RoboCup-Umgebung.

5.2 Goaltracking

Wie in Kapitel 3.2.3 beschrieben, führt Goaltracking eine Normierung der eingehenden Aktivierung durch, indem nur die stärkste Verbindung zu jedem Ziel berücksichtigt wird. Dadurch wird die Bevorzugung appetitiver Verhalten durch Aktivierungsrückkopplung und Aktivierungszusammenfluß verhindert (Kapitel 2.5.2). Daher ist zu erwarten, daß Goaltracking - besonders bei großen Netzwerken mit starker Vernetzung - bessere Ergebnisse liefert, da besonders in solchen Netzwerken Probleme wie Rückkopplung und Aktivierungsverteilung auftreten. Weiterhin sollte Goaltracking in solchen Domänen von Vorteil sein, in denen zielgerichtetes Verhalten besonders wichtig ist.

Um den Einfluß des Goaltracking auf die Verhaltensauswahl zu untersuchen, wurden Netzwerke mit Goaltracking mit Netzwerken ohne Goaltracking verglichen. Die Netzwerke ohne Goaltracking entsprachen dabei Variation drei von [Tyrrell, 1994]. Eingehende Aktivierung wurde immer aufsummiert anstatt nur die maximale Zielaktivierung zu berücksichtigen (Gl. 3.5).

5.2.1 Blockwelt

Der Vergleich von Netzwerken mit Goaltracking mit Netzwerken ohne Goaltracking wurde in der statischen Blockwelt-Domäne durchgeführt. Diese erfordert ausschließlich zielgerichtetes Verhalten, da sie statisch ist und somit dem Agenten beliebig viel Zeit für eine Entscheidung läßt. Ebenso kann der Einfluß von Goaltracking auf größere Netzwerke untersucht werden. Ein Netzwerk für ein Blockwelt-Problem mit vier Blöcken besitzt z.B. vier Ziele, 32 Verhalten, 25 Wahrnehmungspropositionen und etwa 650 Verbindungen. Damit liegt die durchschnittliche Anzahl Aktivierungsverbindungen bei ca. 20 Verbindungen je Kompetenzmodul. Entsprechend hoch ist die Wahrscheinlichkeit, daß Aktivierungsrückkopplung und Aktivierungszusammenfluß auftritt. Verhaltensnetzwerke mit Goaltracking sollten also in dieser Domäne das Ziel zuverlässiger erreichen.

Da die Blockwelt Domäne statisch ist, also kein Zeitdruck bei der Verhaltensauswahl herrscht, wurde die Verhaltensauswahl dahingehend geändert, daß solange Aktivierung ausgetauscht wurde, bis die Aktivierung einen stabilen Zustand erreichte. Dann wurde das aktivste ausführbare Verhalten ausgeführt. Dadurch wird die maximale Antizipationsweite erreicht (siehe Kapitel 3.2). Die Aktivierungsschwelle θ wurde ignoriert, und die Trägheit der Aktivierung wurde abgeschaltet ($\beta = 0$). Dadurch ist die Handlungskontrolle in diesen Versuchen nur noch von den beiden Parametern γ und δ abhängig.

Die Aufgabe bestand darin, einen Turm mit Blöcken A-B-C-D herzustellen

aus der Anfangssituation C, D auf dem Tisch, B auf C und A im Greifer (Abb. 4.2). Schwierigkeit der Aufgabe ist es, das naheliegende Ziel Block A direkt auf B zu legen zu unterdrücken, da zuerst noch die Blöcke B und C auf Block D liegen müssen. Für die Untersuchungen wurden die beiden verbleibenden Parameter γ und δ in ihrem Definitionsbereich mit einer Schrittweite von 0.1 variiert, woraus sich 121 Versuche ergaben, die Aufgabe zu lösen[1].

In Abbildung 5.2 sind die Ergebnisse dargestellt. Das Verhaltensnetzwerk mit Goaltracking konnte bei etwa einem Viertel der Parametervariationen die Aufgabe lösen. Ohne Goaltracking wurde das Ziel der Aufgabe nur bei einer Parametereinstellung erreicht. Das entsprechende Problem mit fünf Blöcken kann ohne Verwendung von Goaltracking mit keiner verwendeten Parametereinstellung mehr gelöst werden. Mit Goaltracking kann es noch in knapp zehn Prozent aller verwendeten Parameterwerte gelöst werden. Goaltracking sorgt also wie erwartet bei den großen Verhaltensnetzwerken der Blockwelt Domäne dafür, daß die Handlungen des Agenten zielgerichtet sind und die Ziele zuverlässiger erreicht werden.

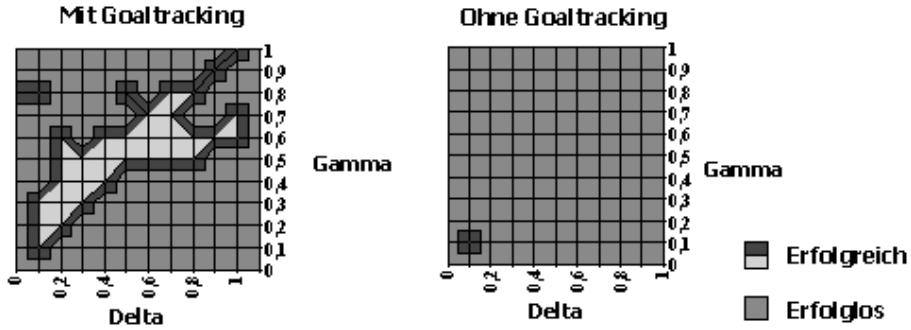

Abbildung 5.2: Vergleich von Goaltracking und Aktivierungssummation (ohne Goaltracking) bei der Lösung einer Blockweltaufgabe mit vier Blöcken.

5.2.2 Fußballumgebung

In einer Serie von 30 Fußballspielen wurden die beiden oben beschriebenen Varianten ebenfalls verglichen. Das erweiterte Verhaltensnetzwerk enthielt vier Ziele, neun Kompetenzmodule, 13 Wahrnehumngspropositionen und ca. 45 Verbindungen. Im Gegensatz zum Verhaltensnetzwerk der Blockwelt ist also hier sowohl die absolute Anzahl Aktivierungsverbindungen als auch die mittlere Anzahl Verbindungen je Kompetenzmodul deutlich geringer. Daher sollte der Einfluß von Goaltracking deutlich schwächer sein, da im Vergleich zum Blockwelt-Netzwerk potentiell weniger Aktivierungsrückkopplung und Aktivierungszusammenfluß im Verhaltensnetzwerk ohne Goaltracking auftreten sollte. Tatsächlich

[1]Da bei den Parameterwerten $\gamma = 1.0$ bzw. $\delta = 1.0$ potentiell Aktivierungsoszillationen auftreten, wurde auch eine maximale Anzahl von Aktivierungszyklen festgelegt, nach der das aktivste Kompetenzmodul ausgeführt wurde. Die Werte wurden verwendet, um den Einfluß der Parameter auch für maximale Parameterwerte untersuchen zu können.

ergab sich kein signifikanter Unterschied zwischen den beiden Mannschaften (Tab. 5.2).

	mit Goaltracking	ohne Goaltracking	p ($n = 30$)
mittlere Anzahl Tore	5.6	6.2	0.28

Tabelle 5.2: Vergleich von Goaltracking und Aktivierungssummation (ohne Goaltracking) in der Fußballumgebung.

5.3 Verhaltensparametrisierung

In Kapitel 3.3.4 wurde beschrieben, wie Verhalten mittels des normierten Ausführungswerts parametrisiert werden können. Dadurch kann die Verhaltensausführung besser an die aktuelle Situation angepaßt werden, da die Intensität der Ausführung eines Verhaltens beeinflußt werden kann.

Zur Untersuchung von Verhaltensparametrisierung in der RoboCup-Umgebung wurde das Verhalten 'zum Ball Rennen' parametrisiert. Bei einem normierten Ausführungswert von 0.0 wurden diese Verhalten mit 60 Prozent Leistung ausgeführt, bei einem Wert von 1.0 mit 100 Prozent. Dazwischen wurde linear interpoliert. Da die Relevanz, ein Tor zu erzielen, mit der Nähe zum Tor, bzw. die Relevanz, ein Gegentor zu verhindern, mit der Nähe zum eigenen Tor steigt, steigt auch der Ausführungswert der Verhalten, die von diesen Zielen Aktivierung erhalten. Der Agent ist aggressiver in der Nähe der Tore und verhaltener im Mittelfeld. Dadurch sollte der Agent Ausdauer in den weniger wichtigen Spielsituationen einsparen, die ihm im Torbereich einen Vorteil verschaffen kann.

5.3.1 Normierung des Ausführungswerts

In Kapitel 3.3.4 wurde die Notwendigkeit der Normierung des Ausführungswerts aufgezeigt, und es wurden zwei Verfahren vorgestellt, wie die Normierung des Ausführungswerts erfolgen kann. Eine Möglichkeit besteht darin, die minimalen und maximalen Ausführungswerte eines Kompetenzmoduls zu speichern und diesen Wertebereich auf das Intervall [0..1] abzubilden (MinMax). Eine weitere Möglichkeit ist es, den mittleren Ausführungswert μ und dessen Standardabweichung s (inkrementell) zu berechnen und einen Bereich von $\mu - k \cdot s$ bis $\mu + k \cdot s$ auf das Intervall [0..1] abzubilden (Verteilung).

Da die MinMax-Normierung anfällig gegenüber besonders hohen oder niedrigen Ausführungswerten ist, ist zu erwarten, daß die Verteilungs-Normierung bessere Ergebnisse liefert. Um das empirisch zu prüfen, wurden 30 Spiele einer Mannschaft mit MinMax-Normierung gegen eine Mannschaft mit Verteilungs-Normierung gespielt. Dabei wurde für die Verteilungs-Normierung $k = 1$ gewählt. Die Mannschaft mit Verteilungs-Normierung erzielte erwartungsgemäß signifikant mehr Tore als die Mannschaft mit MinMax-Normierung (Tabelle 5.3).

	MinMax	Verteilung	p ($n = 30$)
mittlere Anzahl Tore	4.2	6.0	0.008

Tabelle 5.3: Vergleich der MinMax- und der Verteilungs-Normierung.

5.3.2 Vergleich dynamisch - statisch

Die situationsangepaßte (dynamische) Intensität der Verhaltensausführung sollte gegenüber einer statischen Intensität die Ausnutzung der Ressource Ausdauer verbessern und so die Gesamtqualität des Verhaltens, die sich in der Anzahl erzielter Tore ausdrückt, steigern. Um zu untersuchen, inwieweit Verhaltensparametrisierung zu besserer Verhaltenskontrolle in der Robocup-Umgebung führt, wurden Spiele einer Mannschaft mit dynamischer Verhaltensparametrisierung gegen eine Mannschaft mit statischer Verhaltensparametrisierung gespielt. Zur Normierung des Ausführungswerts wurde die Verteilungs-Normierung gewählt. Der Wert des statischen Verhaltensparameters wurde im Definitionsbereich variiert, um dynamische Parametrisierung mit beliebigen statischen Parametern vergleichen zu können. Zu erwarten ist, daß für niedrige Parameterwerte der statischen Mannschaft der Nachteil, langsamer am Ball zu sein, schwerer wiegt als der Vorteil der geringeren Ermüdung. Für hohe Parameterwerte der statischen Mannschaft sollte der Nachteil, schneller zu ermüden, den Vorteil, schneller am Ball zu sein, überwiegen.

Die Mannschaft mit dynamisch parametrisiertem Verhalten erzielte für alle Variationen des statischen Parameters signifikant mehr Tore als die Mannschaft mit statischer Parametrisierung (Abb. 5.3).

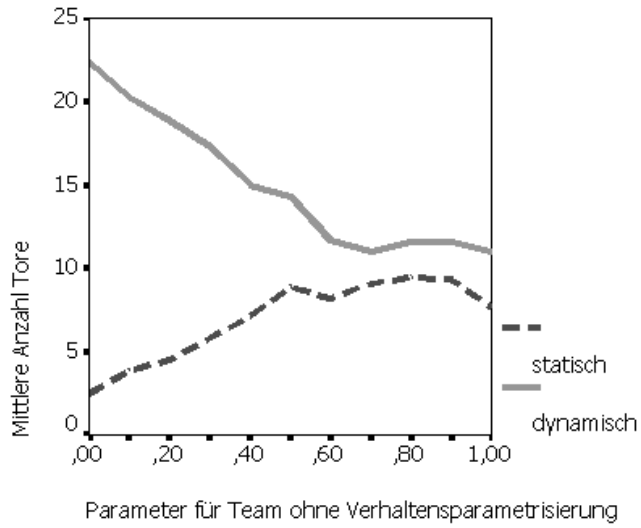

Abbildung 5.3: Mittlere Anzahl Tore bei statischer und dynamischer Verhaltensparametrisierung.

Aufschlußreich ist der Vergleich der mittleren Anzahl Erholungspausen beider Mannschaften. Sie ist ein Maß für den Verbrauch an Ausdauer eines Agenten. Bei der Mannschaft mit statischer Verhaltensparametrisierung wächst sie erwartungsgemäß mit steigenden Parameterwerten (Abb. 5.4). Interessant ist

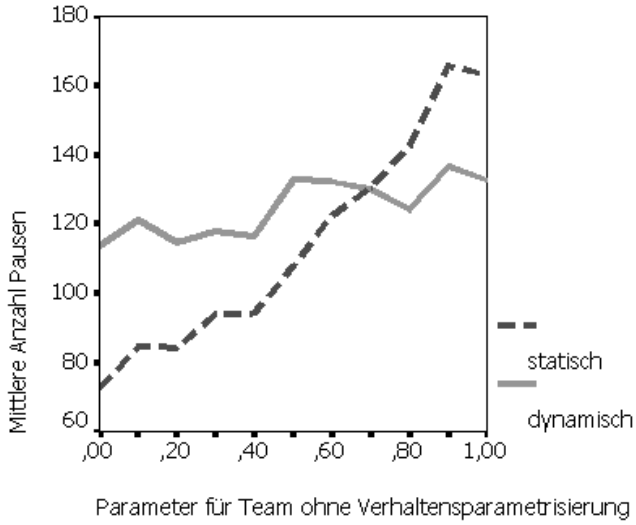

Abbildung 5.4: Mittlere Anzahl Erholungspausen bei statischer und dynamischer Verhaltensparametrisierung.

der Vergleich am Schnittpunkt beider Kurven beim Parameterwert von 0.7. Obwohl beide Mannschaften sich im Mittel gleich oft erholen müssen, erzielte die Mannschaft mit dynamischer Verhaltensparametrisierung signifikant mehr Tore (Tab. 5.4). Der mittlere Verbrauch der Ressource Ausdauer beider Mannschaften ist gleich. Die Mannschaft mit dynamischer Verhaltensparametrisierung verbraucht aber dann mehr Ressourcen, wenn der erwartete Nutzen des Verhaltens zum Ball zu rennen höher ist. Die situationsangepaßte Intensität der Verhaltensausführung führt zu einem besseren Ergebnis.

Parameter statische Mannschaft = 0.7	statisch	dynamisch	p ($n = 45$)
mittlere Anzahl Tore	8.9	11.2	0.003
mittlere Anzahl Pausen	130.6	130.2	0.950

Tabelle 5.4: Vergleich der mittleren Anzahl Tore und Erholungspausen bei statischer (Parameter = 0.7) und dynamischer Verhaltensparametrisierung.

5.4 Reellwertige Propositionen

Reellwertige Propositionen wurden eingeführt, um die aktuelle Situation auch in kontinuierlichen Umgebungen so genau wie möglich erfassen zu können. Sie sorgen dafür, daß die Ausführbarkeit und Aktivierungsweitergabe von Kompetenzmodulen sowie die Relevanz von Zielen kontinuierliche Werte annehmen können (Kapitel 3.3.3). Dadurch sollte sich die Qualität der Handlungskontrolle in kontinuierlichen Umgebungen verbessern. Dieser Einfluß auf Kompetenzmodule und Ziele wurde im Vergleich zu Verhaltensnetzwerken mit binärer τ_P-Funktion untersucht. Zuerst wurde der Einfluß verschiedener und-Verknüpfungen auf die Qualität der Verhaltenskontrolle verglichen.

5.4.1 Und-Verknüpfung

Wie in Kapitel 3.1 beschrieben ist $\tau_P(p \wedge q, s) := \tau_P(p, s) \otimes \tau_P(q, s)$ und $\otimes$ eine beliebige kontinuierliche T-Norm, d.h. ein kommutativer, assoziativer, in jedem Argument monoton wachsender Operator mit eins als neutralem Element [Saffiotti *et al.*, 1995]. Beispiele sind die Minimum-Funktion $min(a, b)$ oder das Produkt $a \cdot b$ zweier Wahrheitswerte τ_P. Die Frage war, ob unterschiedliche und-Verknüpfungen einen signifikanten Einfluß auf die Qualität der Handlungskontrolle haben.

Um den Einfluß verschiedener T-Normen auf die Qualität der Handlungskontrolle zu untersuchen, wurden Spielserien durchgeführt, bei denen die Agenten beider Mannschaften unterschiedliche Funktionen verwendeten. Neben der Minimum-Funktion und dem Produkt wurde noch eine dritte Verknüpfung eingeführt mit $a \otimes b := \sqrt{a \cdot b}$ [2] (normiertes Produkt). Kompetenzmodule mit mehreren Vorbedingungen werden so gegenüber solchen mit wenigen Vorbedingungen bevorzugt. Die Ausführbarkeit eines Kompetenzmoduls wird durch das Ziehen der n-ten Wurzel (n = Anzahl Vorbedingungen) quasi normiert.

Die Ergebnisse der Untersuchungen sind in Tabelle 5.5 dargestellt. Einziges signifikantes Ergebnis lieferte der Vergleich zwischen Produkt und normiertem Produkt, bei dem Agenten mit dem normierten Produkt im Mittel 2.8 Tore mehr erzielten als Agenten mit einfachem Produkt. Daher wurde in allen folgenden Versuchen mit reellwertigen Propositionen $min(a, b)$ für die $\otimes$-Verknüpfung verwendet.

	Tore Team1	Tore Team2	p ($n = 30$)
$min(a, b)$ gegen $a \cdot b$	7.6	6.9	0.35
$min(a, b)$ gegen $\sqrt{a \cdot b}$	7.9	8.2	0.64
$a \cdot b$ gegen $\sqrt{a \cdot b}$	5.7	8.5	< 0.001

Tabelle 5.5: Vergleich verschiedener und-Verknüpfungen für reellwertige Propositionen.

[2] Die Tatsache, daß $\sqrt{a \cdot b}$ nicht assoziativ ist war insofern unproblematisch, als die Assoziativität in der konkreten Implementierung nie in Anspruch genommen wurde

5.4.2 Vergleich mit binären Propositionen

Um zu untersuchen, ob reellwertige Propositionen eine Verbesserung der Qualität der Handlungskontrolle bewirken wurden direkte Vergleiche von Verhaltensnetzwerken mit binären Propositionen und solchen mit reellwertigen Propositionen durchgeführt. Binäre Propositionen wurden aus den reellwertigen Propositionen berechnet, indem die zugehörigen τ_P-Werte bis zu einer Rundungsgrenze rg nach 0 abgerundet und über dieser Rundungsgrenze nach 1 aufgerundet wurden. Die Rundungsgrenze selbst wurde systematisch variiert, um auch deren Einfluß zu untersuchen. Für jeden Wert von rg wurden 60 Spiele gespielt. Beide Mannschaften verwendeten $min(a, b)$ als und-Verknüpfung, sowie Verhaltensparametrisierung und Zielrelevanz.

Die Ergebnisse sind in Abbildung 5.5 dargestellt. Das Schaubild zeigt, daß für einige Rundungswerte kein signifikanter Unterschied zwischen den beiden Mannschaften besteht, daß also bei entsprechender Optimierung des Rundungswerts kein signifikant besseres Ergebnis von reellwertigen Propositionen erzielt wurde.

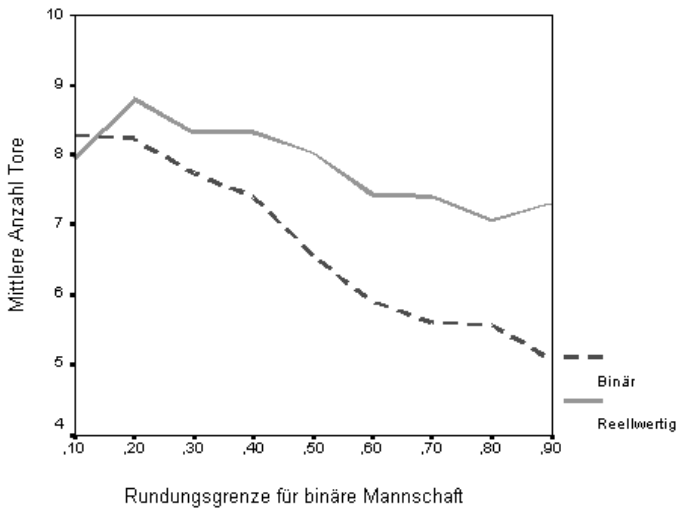

Abbildung 5.5: Vergleich von binären und reellwertigen Propositionen bei unterschiedlicher Rundungsgrenze für die binären Propositionen.

Eine Ursache hierfür wird ersichtlich, wenn man die Verteilung der τ_P-Werte betrachtet (Tab. 5.6). Wie deutlich zu sehen ist, sind die verwendeten Propositionen, bzw. die τ_P-Funktion nur bedingt für den Vergleich mit binären Propositionen geeignet , da Werte zwischen null und eins nur in maximal acht Prozent aller Fälle auftreten. Der Unterschied zwischen beiden Mannschaften ist gering.

Daher wurden in einer zweiten Untersuchung drei neue Propositionen 'lowStamina', 'ballIsNearOwnGoal' und 'ballIsNearOtherGoal' eingeführt. Der Wahrheitswert τ_P von 'lowStamina' war über 2000 stamina (50% Ausdauer) null und ohne jegliche Ausdauer eins mit linearer Interpolation dazwischen. Der Wahrheitswert für 'ballIsNearOwnGoal' war eins, wenn der Ball maximal 15 m vom

Proposition	$\tau_P = 0$	$0 < \tau_P < 1$	$\tau_P = 1$
ballKickable	91.5	1.9	6.6
haveSeenBall	3.0	4.2	92.8
nearOtherGoal	91.6	8.4	0.0
nearOwnGoal	77.8	7.0	15.2
opponentIsNearerToBall	29.4	5.6	65.0
opponentIsNearerToOwnGoal	53.7	3.4	42.9
teammateIsUnmarked	89.0	2.4	8.6
teammateIsNearerBall	49.7	2.9	47.4
teammateIsNearerOtherGoal	60.2	1.9	37.9

Tabelle 5.6: Verteilung der τ_P-Werte während eines Spiels (in Prozent aller Werte).

eigenen Tor entfernt war und null bei mehr als 70 m Entfernung. Entsprechendes gilt für 'ballIsNearOtherGoal' und dem gegnerischen Tor. D.h. in einem weiten Bereich konnten Wahrheitswerte dieser Propositionen zwischen null und eins angenommen werden. Die resultierende Verteilung von τ_P-Werten ist in Tabelle 5.7 dargestellt. Mehr als die Hälfte aller Wahrheitswerte dieser Propositionen liegen zwischen null und eins.

Proposition	$\tau_P = 0$	$0 < \tau_P < 1$	$\tau_P = 1$
ballIsNearOtherGoal	31.0	63.5	5.5
ballIsNearOwnGoal	18.1	73.5	8.4
lowStamina	19.5	80.0	0.5

Tabelle 5.7: Verteilung der τ_P-Werte für neu eingeführte Propositionen (in Prozent aller Werte).

Das Ziel, ein Tor zu erzielen, erhielt die Relevanzbedingung 'ballIsNearOtherGoal' und das Ziel, Gegentore zu vermeiden, erhielt die Relevanzbedingung 'ballIsNearOwnGoal'. Zusätzlich wurde eine neues Ziel eingeführt 'not lowStamina' mit der Relevanzbedingung 'lowStamina'. Je niedriger also die Ausdauer war, desto stärker wurden Verhalten gehemmt, die Ausdauer verbrauchen, und desto stärker wurde das Verhalten aktiviert, sich auszuruhen. Neben den Zielen, ein Tor zu erzielen, Gegentore zu verhindern und anspielbar zu sein, wurde somit noch ein Ziel eingeführt, das in Konflikt zu diesen steht. Dadurch sollte insbesondere der Einfluß von reellwertigen Propositionen auf die Relevanz von Zielen und die Konfliktresolution zwischen den Zielen verstärkt werden. Gleichzeitig wurde die maximale Ausdauer der Spieler verringert, um den Nutzen einer guten Ausdauerkontrolle zu erhöhen.

Die Ergebnisse von 30 Spielen mit diesen Netzwerken sind in Abbildung 5.6 zusammengefaßt. Die Mannschaft mit reellwertigen Propositionen erzielte in diesen Untersuchungen über weite Bereiche des Rundungswerts signifikant

bessere Ergebnisse als die Mannschaft mit binären Propositionen. Tendentiell signifikant waren die Ergebnisse für die Rundungswerte 0.3 und 0.4.

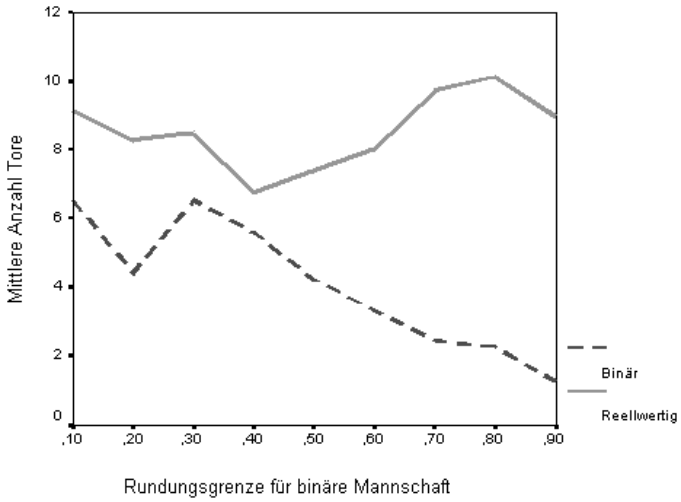

Abbildung 5.6: Vergleich von binären und reellwertigen Propositionen. Im Unterschied zum vorigen Vergleich hatte jeder Spieler eine geringere maximale Ausdauer und das zusätzliche Ziel, genügend Ausdauer zu besitzen.

Reellwertige und binäre Propositionen wurden auch bei Spielen mit elf Spielern mit Hilfe der im RoboCup99 verwendeten Verhaltensnetzwerke verglichen. Die Ergebnisse sind in Abbildung 5.7 dargestellt. Durch die zusätzliche Einführung von reellwertigen Propositionen und Zielen mit Relevanzbedingung ist die mittlere Tordifferenz hier deutlich größer als bei den vorigen Spielen.

5.5 Situationsabhängige Ziele

Ziele in erweiterten Verhaltensnetzwerken können dynamisch priorisiert werden (Kapitel 6.1.3). Dadurch ist der Einfluß von wichtigen Zielen nur in relevanten Situtationen dominant. Weniger wichtige Ziele können das Verhalten bestimmen, solange die wichtigeren Ziele nicht relevant sind. Dies sollte sich, neben einer besseren Qualität der Handlungskontrolle insgesamt, in einem häufigeren Verfolgen weniger wichtiger Ziele ausdrücken.

In einer Serie von 30 Spielen wurde der Einfluß der Relevanzbedingung von Zielen auf das Verhalten von Fußballagenten untersucht. Dazu spielte eine Mannschaft mit statischen Zielen, d.h. mit Zielen ohne Relevanzbedingung, deren Relevanz konstant eins ist, gegen eine Mannschaft mit Relevanzbedingungen und daher dynamischen Zielen. Die beiden verwendeten Verhaltensnetzwerke unterschieden sich also lediglich in den Relevanzbedingungen der Ziele. Das verwendete Verhaltensnetzwerk für statische Ziele entsprach dabei dem bei der Parameteroptimierung verwendeten Netzwerk. Das Verhaltensnetzwerk für dynamische Ziele wurde durch **Hinzufügen** von Relevanzbedingungen aus dem Netzwerk mit statischen Zielen gewonnen. Es wurde also nicht ein Verhaltens-

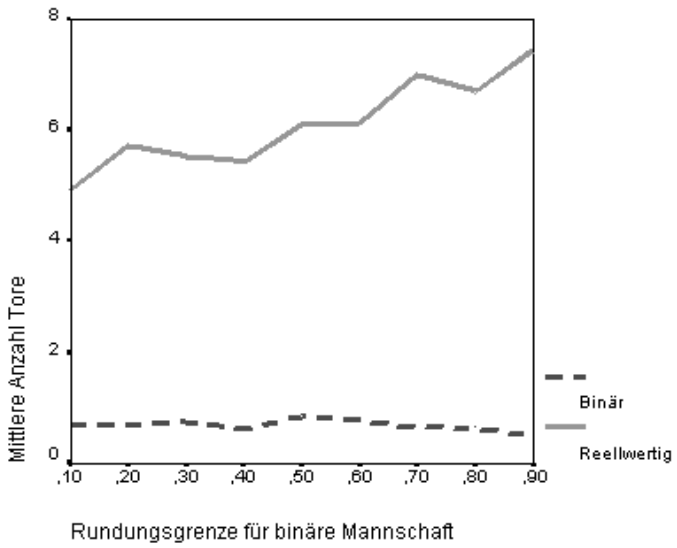

Abbildung 5.7: Vergleich von binären und reellwertigen Propositionen bei 11 Spielern

netzwerk für dynamische Ziele optimiert und zum Vergleich mit statischen Zielen die Relevanzbedingungen entfernt, sondern zu einem funktionierenden und optimierten Verhaltensnetzwerk mit statischen Zielen wurden Relevanzbedingungen hinzugefügt. Die erstgenannte Vorgehensweise wurde in einer zweiten Untersuchung angewandt.

Das verwendete Netzwerk enthielt drei Ziele:

- 'goal', ein Tor zu erzielen mit Wichtigkeit 1.0 und Relevanzbedingung 'inOtherHalf', die 20 m hinter der Mittellinie wahr und 20 m vor der Mittellinie falsch war mit linearer Interpolation dazwischen.
- 'protectOwnGoal', ein Gegentor zu vermeiden mit Wichtigkeit 0.9 und Relevanzbedingung 'not inOtherHalf'.
- 'beUnmarked', frei zu stehen, um anspielbar zu sein mit Wichtigkeit 0.6 und Relevanzbedingung 'teammateHasBall', die wahr war, wenn der Mitspieler der am nächsten zum Ball stehende Spieler war.

Die Ergebnisse der Untersuchung sind in Tabelle 5.8 dargestellt. Die Mannschaft mit dynamischen Zielen erzielte signifikant mehr Tore als die Mannschaft mit statischen Zielen. Auffälligster Unterschied zwischen den beiden Mannschaften ist der signifikante Unterschied in der Anzahl Verhaltensausführungen 'zum Ball laufen' und 'sich frei laufen'. Die Agenten mit Zielrelevanz liefen seltener zum Ball und liefen sich dafür öfter frei. Die Ursache hierfür ist, daß im Verhaltensnetzwerk mit statischen Zielen das weniger wichtige Ziel 'beUnmarked' von den beiden anderen, wichtigeren Zielen dominiert wird. Bei den Verhaltensnetzwerken mit dynamischen Zielen hat in solchen Situationen, in denen das Erzielen von Toren und das Vermeiden von Gegentoren weniger relevant ist, das Ziel

frei zu stehen einen verhältnismäßig großen Einfluß, wodurch das Verhalten, sich frei zu laufen, öfter ausgeführt wird. Entsprechend häufiger wurde der Ball abgespielt.

	statische Zielrelevanz	dynamische Zielrelevanz	p ($n = 30$)
Mittlere Anzahl Tore	6.6	10.8	< 0.001
zum Ball laufen	2857	1485	< 0.001
sich frei laufen	631	2206	< 0.001
abspielen	30	64	< 0.001

Tabelle 5.8: Vergleich von statischen und dynamischen Zielen bei Spielen mit zwei Spielern.

Auch für die Verhaltensnetzwerke für Spiele mit 11 Spielern wurde ein Vergleich von statischen und dynamischen Zielen durchgeführt. Verwendet wurden dabei die im RoboCup99 eingesetzten Verhaltensnetzwerke mit der einen Änderung, daß der Torwart den Ball nicht fangen konnte. Dadurch wurde gewährleistet, daß genügend Tore erzielt werden konnten. Im Gegensatz zu obiger Untersuchung wurden hier also Verhaltensnetzwerke untersucht, die auf die Verwendung von Relevanzbedingungen optimiert waren. Die Verhaltensnetzwerke mit statischen Zielen gingen aus denen mit dynamischen Zielen durch **Entfernen** der Relevanzbedingungen hervor.

Tabelle 5.9 zeigt die Ergebnisse. Die Mannschaft mit dynamischen Zielen erzielte im Schnitt über fünf Tore mehr als die Mannschaft mit statischen Zielen. Der Unterschied fällt entsprechend der Versuchsbedingungen erwartungsgemäß höher aus als in der ersten Untersuchung.

	statische Zielrelevanz	dynamische Zielrelevanz	p ($n = 30$)
Mittlere Anzahl Tore	1.3	6.7	< 0.001

Tabelle 5.9: Vergleich von statischen und dynamischen Zielen bei Spielen mit elf Spielern.

5.6 Parallele Ausführung von Verhalten

Kapitel 3.3.1 beschreibt, wie mehrere Verhalten in erweiterten Verhaltensnetzwerken gleichzeitig ausgeführt werden können, wenn sie unterschiedliche Ressourcen verwenden. Dadurch können Ziele schneller erreicht werden oder mehrere Ziele gleichzeitig verfolgt werden. Besonders in dynamischen Umgebungen sollte das die Qualität der Handlungskontrolle verbessern.

Mit der beim RoboCup99 verwendeten neuen Version 5 des `soccerservers` wurde es möglich, die parallele Auswahl und Ausführung von Verhalten erweiterter Verhaltensnetzwerke zu untersuchen. Der neue Server erlaubt unabhängig und gleichzeitig zum Körper den Kopf um 90° nach links und rechts zu drehen. Dadurch ist es z.B. möglich, sich freizulaufen und trotzdem den Ball im Auge zu behalten. Eine gleichzeitige Kopfdrehung und Körperbewegung spart dabei Zeit gegenüber einer seriellen Ausführung. Ebenfalls gleichzeitig zu Bewegungen des Agenten kann der Agent sprechen. Um die Bandbreite der Kommunikation menschlichen Verhältnissen anzunähern, kann nur ein (beliebiger) Spieler einer Mannschaft alle zwei Zyklen (200ms) etwas sagen. Dadurch kann ein konkreter Spieler nur in etwa vier Prozent aller Zyklen sprechen.

Die Agenten der MagmaFreiburg Mannschaft drehen den Kopf immer dann Richtung Ball, wenn der Ball das Gesichtsfeld verlassen hat. Dadurch kann das Gesichtsfeld relativ zur Bewegungsrichtung von 90° ohne Kopfdrehung auf 270° mit Kopfdrehung vergrößert werden. Wie in Kapitel 4.2 beschrieben, wurde Kommunikation eingesetzt, um das globale Weltmodell zu verbessern. Alle zwei Zyklen teilt jeweils ein anderer Spieler die von ihm wahrgenommenen Positionen anderer Spieler und des Balls den anderen Agenten mit. Beide Verhalten führten zusammen mit den sonstigen Bewegungs- und Dribblingverhalten dazu, daß in weniger als zwei Prozent aller Zyklen mehr als eine Aktion gleichzeitig ausgeführt wurde.

Für den Vergleich von serieller und paralleler Verhaltensausführung wurde eine Mannschaft mit paralleler mit einer Mannschaft mit serieller Verhaltensausführung verglichen. Die beiden Mannschaften waren identisch und entsprachen der im RoboCup99 verwendeten Mannschaft. Einziger Unterschied war, daß bei der Mannschaft mit serieller Verhaltensausführung die Verhaltensauswahl im aktuellen Zyklus nach der Selektion und Ausführung des aktivsten Verhaltens beendet wurde. Die Mannschaft mit paralleler Verhaltensausführung erzielte signifikant mehr Tore als die Mannschaft mit serieller Ausführung (Tab. 5.10), obwohl nur in etwa zwei Prozent aller Zyklen Verhalten parallel ausgeführt wurden.

	seriell	parallel	p ($n = 30$)
Mittlere Anzahl Tore	2.4	4.3	< 0.001

Tabelle 5.10: Vergleich von serieller und paralleler Verhaltensausführung.

5.7 Vergleich mit MASM

Da erweiterte Verhaltensnetzwerke auf MASM-Netzwerken basieren, bietet sich ein Vergleich beider Handlungskontrollmechanismen an. Dieser abschließende Vergleich dient im Wesentlichen dazu, die Qualität der Verbesserungen und Erweiterungen erweiterter Verhaltensnetzwerke gegenüber dem Basis-Mechanismus einschätzen zu können.

Für den Vergleich von erweiterten Verhaltensnetzwerken mit MASM-Netzwerken ist es zunächst notwendig, den Prozeß der Parameteroptimierung (Kapitel 5.1) auch für MASM-Netzwerke durchzuführen. Aufgrund der unterschiedlichen Aktivierungs- und Selektionsmechanismen sind für MASM-Netzwerke gegenüber erweiterten Verhaltensnetzwerken abweichende Parametereinstellungen zu erwarten. Die im Prozeß der Parameteroptimierung gefundenen Parameterwerte sind in Tabelle 5.11 wiedergegeben.

ϕ	γ	δ	π	θ
0.0	0.3	0.5	1.8	1.8

Tabelle 5.11: Verwendete Parameter für MASM-Netzwerke in der RoboCup-Umgebung.

Um die Wahrscheinlichkeit von Implementierungsfehlern auf ein Minimum zu reduzieren, sind sowohl MASM-Netzwerke als auch erweiterte Verhaltensnetzwerke abgeleitete Klassen einer gemeinsamen Basisklasse 'Verhaltensnetzwerk'. Diese enthält alle generell für Verhaltensnetzwerke benötigte Funktionalität, während die abgeleiteten Klassen nur die mechanismusspezifische Funktionalität implementieren. Dadurch sind nur etwa 20 Prozent des Codes der MASM-Netzwerke abweichend von der gemeinsam genutzten Funktionalität der Basisklasse. Die grafische Echtzeit-Darstellung der Netzwerke erlaubte zudem eine sehr effiziente Funktionskontrolle und Fehlersuche, wodurch Einflüsse der konkreten Implementierung auf die Versuchsergebnisse nicht vorhanden sein sollten.

Zunächst wurde in einer Serie von 30 Fußballspielen der geänderte Aktivierungsmechanismus von erweiterten Verhaltensnetzwerken mit dem ursprünglichen, in MASM-Netzwerken verwendeten verglichen. Erweiterungen wie reellwertige Propositionen, situationsabhängige Ziele oder die gleichzeitige Ausführung von Verhalten wurden nicht verwendet. Die Ergebnisse sind in Tabelle 5.12 dargestellt. Die Mannschaft mit erweiterten Verhaltensnetzwerken erzielte im Mittel mehr Tore als die MASM-Mannschaft. Auffälligster Unterschied beider Mannschaften war die signifikant höhere Anzahl Verhaltenswechsel der MASM-Mannschaft. Hauptgrund ist die Rücksetzung der Aktivierung des ausgeführten Moduls bei MASM. Diese ist sinnvoll in diskreten Umgebungen, in denen nach einmaliger Ausführung eines Verhaltens die Effekte sofort eintreten. In kontinuierlichen Umgebungen ist es aber oft notwendig, ein Verhalten längere Zeit auszuführen, um die gewünschten Effekte zu erzielen (z.B. zum Ball rennen).

	MASM	EBN Aktivierung	p ($n = 30$)
Mittlere Anzahl Tore	6.3	7.8	0.03
Verhaltenswechsel	1105	546	< 0.001

Tabelle 5.12: Vergleich von MASM mit dem EBN Aktivierungsmechanismus.

Wird die Aktivierung des ausgeführten Moduls in MASM nicht zurückgesetzt (Tab. 5.13), finden sogar weniger Verhaltenswechsel als bei erweiterten Verhaltensnetzwerken statt. Letztere erzielen aber trotzdem im Mittel mehr Tore.

	MASM ohne Aktivierungsrücksetzung	EBN Aktivierung	p ($n = 30$)
Mittlere Anzahl Tore	4.6	5.6	0.04
Verhaltenswechsel	376	540	< 0.001

Tabelle 5.13: Vergleich von MASM ohne Aktivierungsrücksetzung mit EBN.

In einer weiteren Serie von 30 Spielen wurden reellwertige Propositionen und situationsabhängige Ziele in den erweiterten Verhaltensnetzwerken verwendet. Die verwendeten Netzwerke entsprachen denen in Kapitel 5.5. Erweiterte Verhaltensnetzwerke erzielten im Mittel signifikant mehr Tore als MASM-Netzwerke (Tab. 5.14).

	MASM	EBN	p ($n = 30$)
Mittlere Anzahl Tore	4.2	10.9	< 0.001

Tabelle 5.14: Vergleich von MASM und EBN mit reellwertigen Propositionen und situationsabhängigen Zielen.

Schließlich wurde auch ein Vergleich zwischen MASM- und erweiterten Verhaltensnetzwerken mit Hilfe der im RoboCup99 verwendeten Verhaltensnetzwerke durchgeführt. Die durchgeführten Spiele waren also Spiele mit elf gegen elf Spielern. Die erweiterten Verhaltensnetzwerke verwendeten reellwertige Propositionen, situationsabhängige Ziele sowie parallele Verhaltensausführung. Die Mannschaft mit erweiterten Verhaltensnetzwerken erzielte auch bei Spielen mit elf Spielern signifikant mehr Tore als die MASM-kontrollierte Mannschaft (Tab. 5.15).

	MASM	EBN	p ($n = 18$)
Mittlere Anzahl Tore	0.1	8.9	< 0.001

Tabelle 5.15: Vergleich von MASM und EBN mit paralleler Verhaltensausführung bei Spielen mit 11 Spielern.

Kapitel 6

Diskussion

In diesem Kapitel wird diskutiert, inwieweit erweiterte Verhaltensnetzwerke den Anforderungen an eine schnelle und zielgerichtete Handlungskontrolle in dynamischen und nicht-deterministischen Umgebungen gerecht werden (siehe Kapitel 1.1.3). Dabei wird auch erläutert, inwieweit erweiterte Verhaltensnetzwerke in der Lage sind, menschliches Entscheidungsverhalten zu reproduzieren. Schließlich werden Einschränkungen erweiterter Verhaltensnetzwerke beschrieben sowie verschiedene vorgeschlagene Lösungsansätze diskutiert.

6.1 Motivation und Zielmanagement

6.1.1 Deskriptive Entscheidungstheorie

Grundlegendes Prinzip vieler Motivationstheorien ist das Erwartungs-Wert-Prinzip. "Es gibt wohl keine neuere Motivationstheorie, die nicht in ihren Grundzügen dem Modelltyp der sog. Erwartungs-Wert-Theorien entspräche." [Heckhausen, 1989]. Empirische Untersuchungen zeigen, daß menschliches Entscheidungsverhalten nicht immer dem Erwartungs-Wert Prinzip der Entscheidungstheorie entspricht. Oft verletzen Menschen die Axiome der Nutzentheorie und treffen beispielsweise intransitive Entscheidungen [Tversky, 1969] oder lassen sich von irrelevanten Informationen beeinflussen [Allais, 1953]. Dementsprechend sind Erwartungs-Wert Modelle zunächst nur bedingt geeignet, tatsächliches menschliches Entscheidungsverhalten zu prognostizieren.

Verschiedene Vorschläge wurden gemacht (siehe auch Kapitel 1.2), das Erwartungs-Wert Modell der Entscheidungstheorie zu erweitern. Die Prospect-Theorie [Kahnemann und Tversky, 1979] gilt als die am meisten beachtete Erweiterung [Jungermann *et al.*, 1998; Eisenführ und Weber, 1999]. Durch die Einführung einer Wahrscheinlichkeitsgewichtungsfunktion werden objektive Wahrscheinlichkeiten nicht-linear auf subjektive Wahrscheinlichkeiten abgebildet. Das trägt Untersuchungen von menschlichem Entscheidungsverhalten Rechnung, bei denen niedrige Wahrscheinlichkeiten (ausgenommen das unmögliche Ereignis) proportional übergewichtet und hohe Wahrscheinlichkeiten (ausgenommen das sichere Ereignis) proportional untergewichtet wurden. Weiterhin zeigen empirische Untersuchungen, daß menschliche Entscheidungen davon abhängen, ob der Effekt als Gewinn oder Verlust interpretiert wird. Menschen

zeigen bei Gewinnen ein eher risiko-aversives Entscheidungsverhalten, im Zusammenhang mit Verlusten aber ein eher risiko-geneigtes Verhalten.

So bevorzugten beispielsweise in einer Untersuchung von Tversky und Kahneman [1981] eine Mehrzahl von Versuchspersonen (VPn) den sicheren Gewinn von 240$ (a) gegenüber einer 25% Chance auf 1000$ (b), obwohl der erwartete Nutzen von a niedriger ist. Umgekehrt wurde ein mit 75% Wahrscheinlichkeit eintretender Verlust von 1000$ (c) einem sicheren Verlust von 750$ (d) vorgezogen, obwohl beide denselben erwarteten Nutzen haben. 73% der VPn wählten die Kombination der Optionen (a) und (c), 3% die Kombination der Optionen (b) und (d). In einer anderen Untersuchung hatten die VPn die Wahl, mit 25% Wahrscheinlichkeit 240$ zu gewinnen und mit 75% Wahrscheinlichkeit 760$ zu verlieren (e) oder mit 25% Wahrscheinlichkeit 250$ zu gewinnen und mit 75% Wahrscheinlichkeit 750$ zu verlieren (f). Erwartungsgemäß wählten alle VPn Option (f). Die Untersuchung ist interessant, weil die von den meisten VPn in der ersten Untersuchung gewählte Kombination der Optionen (a) und (c) der Option (e) und die wenig gewählte Kombination der Optionen (b) und (d) der Option (f) der zweiten Untersuchung entsprechen.

Im Rahmen der Prospect-Theorie wird dem Rechnung getragen, indem die Nutzenfunktion den objektiven Wert einer Option nicht linear auf den subjektiven Nutzen abbildet (siehe Abb. 6.1). Zudem werden die Konsequenzen relativ zum vorher bestimmten Referenzpunkt bewertet, d.h. die Wertfunktion verschiebt sich, je nachdem ob eine Option als Gewinn oder Verlust empfunden wird.

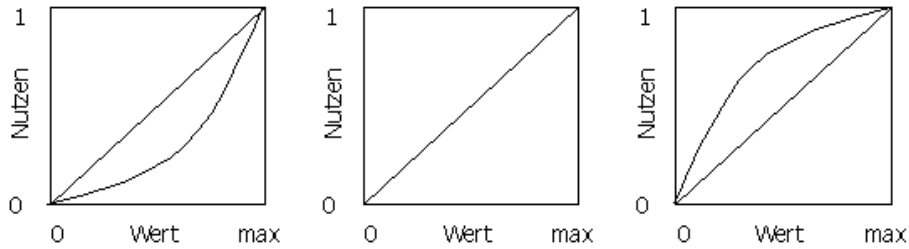

Abbildung 6.1: Nutzenfunktion menschlicher Entscheider mit verschiedenen Risikoeinstellungen: risiko-geneigt (links), risiko-neutral (mitte) und risiko-aversiv (rechts) [Jungermann *et al.*, 1998].

Auch erweiterte Verhaltensnetzwerke können diese Befunde reproduzieren, wenn eine nicht-lineare Nutzenfunktion u verwendet wird. Die Risikoeinstellung von Menschen spiegelt sich in der Stärke und Art der Krümmung der Nutzenfunktion wieder. Diese Krümmung wird durch das Arrow-Prattsche absolute Risikoeinstellungsmaß r(x) definiert [Eisenführ und Weber, 1999]:

$$r(x) = \frac{u''(x)}{u'(x)} \tag{6.1}$$

Friend und Blume [1975] zeigten anhand von US-amerikanischen Steuerdaten, daß Anleger im Durchschnitt abnehmende absolute Risikoeinstellung und kon-

stante relative Risikoeinstellung $x \cdot r(x)$ besitzen. Die Nutzenfunktion

$$u(x) = \begin{cases} x^{2\rho} & : \quad x \geq 0 \\ -x^{2\rho} & : \quad x < 0 \end{cases} \tag{6.2}$$

entspricht diesen Forderungen, falls x einen normierten Wert bezeichnet und $\rho \in]0..1]$ den Risikoparameter darstellt. Für $\rho = \frac{1}{2}$ geht die Nutzenfunktion in eine lineare Nutzenfunktion über und entspricht dem risiko-neutralen Entscheidungsverhalten des Erwartungs-Wert-Modells. Für $\rho < \frac{1}{2}$ ist das Entscheidungsverhalten risiko-aversiv im Fall von Gewinnen ($x \geq 0$) und risiko-geneigt im Fall von Verlusten ($x < 0$). Für $\rho > \frac{1}{2}$ ist es entsprechend umgekehrt. Unter Verwendung dieser Nutzenfunktion konnten mit erweiterten Verhaltensnetzwerken verschiedene empirische Befunde menschlichen Entscheidungsverhaltens, wie der oben beschriebene (siehe Abb. 6.2) und andere in [Jungermann *et al.*, 1998] aufgeführte Befunde, reproduziert werden.

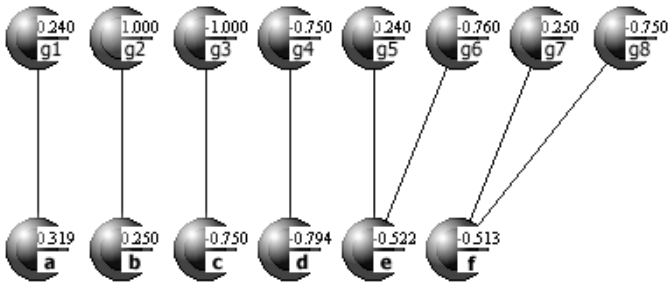

Abbildung 6.2: Erweitertes Verhaltensnetzwerk des oben beschriebenen Entscheidungsproblems. Oben sind die Ziele und deren normierter Wert dargestellt, unten die alternativen Optionen a-f und deren erwarteter Nutzen. Wie die menschlichen VPn ist bei risiko-aversiver Parametrisierung ($\rho = 0.4$) der subjektive erwartete Nutzen von Option a höher als b, c höher als d und e niedriger als f.

6.1.2 Multiple Ziele

Die Belohnungsfunktion eines Markov-Entscheidungsprozesses (MDP) gibt den Wert jedes Zustands der Umgebung an. Die Anzahl Zustände und damit die (Vektor-)Repräsentation der Belohnungsfunktion wächst mit der Zahl der Propositionen exponentiell [Boutilier *et al.*, 1999]. Der Wert eines Zustands ist jedoch in vielen Domänen nicht von der gesamten Zustandsbeschreibung, sondern von wenigen Eigenschaften eines Zustands abhängig. Im Fußball ist z.B. der Wert eines Tors nicht von der Farbe des Balls oder der Größe des Torschützen abhängig. Ziele generalisieren über eine Menge von Zuständen und bilden so eine kompakte Repräsentation einer Belohnungsfunktion. Dadurch kann in einer strukturierten Umgebung die Belohnungsfunktion mit einer relativ geringen Anzahl von Zielen repräsentiert werden. Besitzt eine Umgebung wenig Struktur, d.h. ist die Belohnungsfunktion für viele oder alle Zustände verschieden,

kann aber schlimmstenfalls die Anzahl der Ziele exponentiell mit der Zahl der Propositionen wachsen.

Die Repräsentation der Belohnungsfunktion durch Ziele macht es notwendig, daß mehrere, auch in Konflikt zueinander stehende Ziele gleichzeitig bei der Verhaltensauswahl berücksichtigt werden. Mehrere Ziele können in erweiterten Verhaltensnetzwerken gleichzeitig verfolgt werden, wenn es Verhalten gibt, deren Effekte mehreren Zielen zuträglich sind. Solche Verhalten werden bevorzugt, da sie die Aktivierung von mehreren Zielen summieren (Gl. 3.6). Aktivierung und Hemmung sorgen auch dafür, daß in Konflikt zueinander stehende Ziele bei der Verhaltensauswahl berücksichtigt werden können. Zwei Ziele stehen in Konflikt zueinander, wenn die Mengen der Situationen beider Ziele disjunkt sind. Durch den Mechanismus der Aktivierung berücksichtigen Kompetenzmodule in erweiterten Verhaltensnetzwerken den erwarteten Nutzen, also Wichtigkeit und Erreichbarkeit der Ziele (Gl. 3.1). Können nicht alle Ziele erreicht werden, werden wichtigere, bzw. sicher erreichbare Ziele bevorzugt.

Die kognitiven Architekturen Act-R und Soar sind nur in der Lage, eine Zielhierarchie zu berücksichtigen [Anderson, 1993; Anderson und Lebiere, 1998; Laird *et al.*, 1987; Rosenbloom *et al.*, 1993]. Die Repräsentation der Ziele in einem Zielstapel verhindert, daß mehrere top-level Ziele gleichzeitig berücksichtigt werden. So können Produktionen nicht bevorzugt werden, die mehreren Zielen zuträglich sind. Bei klassischer Handlungsplanung können mehrere Zielbedingungen spezifiziert werden. Diese müssen allerdings konsistent sein, d.h. es wird davon ausgegangen, daß alle Zielbedingungen gleichzeitig erreichbar sind. Handlungsplaner berücksichtigen nicht die unterschiedliche Wichtigkeit von Zielen und bewerten dementsprechend auch die teilweise Erreichung von Zielen als Fehlschlag (siehe auch [Haddawy und Hanks, 1998]).

6.1.3 Dynamische Ziele

Die Wichtigkeit eines Ziels erlaubt die statische Priorisierung der Ziele. Kompetenzmodule, die wichtige Ziele erreichen, erhalten mehr Aktivierung (Gl. 3.1) als solche, die weniger wichtige Ziele wahr machen können und werden bei der Verhaltensauswahl bevorzugt. Die Wichtigkeit allein ist jedoch ein suboptimales Maß für den Einfluß eines Ziels auf die Verhaltensauswahl [Sloman, 1987], [Norman, 1997], [Dörner, 1999]. Wichtige Ziele sind nicht in jeder Situation relevant. So ist das Ziel, beim Fußball ein Tor zu erzielen, sicher ein sehr wichtiges Ziel. Liegt der Ball auf der Torlinie des eigenen Tors, ist es in dem Moment aber nicht relevant. Neben der statischen Priorisierung der Ziele ist es wünschenswert, auch eine situationsabhängige Priorisierung der Ziele zu erlauben.

Daher wurde die Relevanzbedingung von Zielen eingeführt. Mit ihrer Hilfe ist es möglich zu spezifizieren, in welchen Situationen ein Ziel relevant ist. Während Franklin et al [1996] den Einfluß von Zielen bei zunehmender (zeitlicher) Dringlichkeit verstärken, können bei erweiterten Verhaltensnetzwerken beliebige Bedingungen für die Relevanz eines Ziels angegeben werden. Damit können erweiterte Verhaltensnetzwerke, im Unterschied zu Markov-Entscheidungsprozessen (MDP) (siehe Kapitel 1.5.2), auch dynamische Belohnungsfunktionen berücksichtigen. Da die Änderung der Relevanz eines Ziels direk-

ten Einfluß auf die Aktivierung von Kompetenzmodulen hat, wird sich der dynamische Nutzen eines Ziels schnell auf die Verhaltensauswahl auswirken. Bei entscheidungstheoretischen Planern eines MDPs macht eine Änderung der Belohnungsfunktion die Berechnung einer neuen optimalen Aktion für jeden Zustand notwendig.

Zusammen mit dem Mechanismus von Aktivierung und Hemmung sowie der Einführung kontinuierlicher Propositionen erlauben Relevanzbedingungen, verschiedene Arten von Zielen zu modellieren:

- Erreichungsziele (achievement goals [Norman und Long, 1995]) sind Ziele, einen momentan nicht gegebenen Zustand zu erreichen. Beispiel ist ein Tor beim Fußball oder der Zielzustand eines Blockwelt-Problems (siehe Kapitel 4.1). Erreichungsziele sind in der Regel um so relevanter, je näher der Agent dem Ziel ist. Das ist in erweiterten Verhaltensnetzwerken dadurch zu erreichen, daß der Wahrheitswert der Relevanzbedingung bei der Annäherung an das Ziel steigt.

- Erhaltungsziele (maintenance goals [Norman und Long, 1995]), also Ziele einen vorhandenen Zustand zu erhalten, werden durch den Mechanismus der Hemmung umgesetzt. Verhalten, die ein Erhaltungsziel zerstören, werden von diesem gehemmt. Beispiel für ein Erhaltungsziel ist, die Batterieladung eines Roboters ausreichend hoch zu halten. Erhaltungsziele sind in der Regel um so relevanter, je weniger erfüllt sie sind. Je niedriger die Batterieladung, desto drängender sollte das Verhalten aktiviert werden, sie wieder aufzuladen. Führt man für das Ziel 'ausreichend Energie' die Relevanzbedingung 'niedrige Batterieladung' ein, deren Wahrheitswert mit fallender Batterieladung steigt, steigt die Relevanz und damit der Einfluß des Ziels auf die Verhaltensauswahl je leerer die Batterie ist. Verhalten, die die Batterie belasten, werden zunehmend gehemmt, das Aufladeverhalten wird zunehmend aktiviert. Ein Erhaltungsziel hat also typischerweise die Form $(GCon, i, \neg GCon \wedge \ldots)$.

Erreichungs- und Erhaltungsziele können auch in MASM-Netzwerken modelliert werden [Maes, 1990a], sie haben jedoch stetigen und statischen Einfluß auf die Verhaltensauswahl, d.h. von ihnen geht immer dieselbe Aktivierung aus, solange ein Erreichungsziel nicht erreicht, bzw. ein Erhaltungsziel erreicht ist.

6.1.4 Gelegenheiten

Mit Hilfe von dynamischen Zielen können auch günstige Gelegenheiten [Hayes–Roth und Hayes–Roth, 1979; Goodwin und Simmons, 1992; Norman, 1997] der momentanen Situation modelliert werden. Gelegenheiten sind Situationen, in denen mit geringem Zusatzaufwand zusätzliche Ziele erreicht werden können. Beispiel für eine Gelegenheit ist, beim Weg von der Arbeit nach Hause beim Bäcker vorbeizugehen, um Brot zu kaufen. Empirische Untersuchungen zeigen, daß Menschen solche Handlungsgelegenheiten besonders effizient speichern können. Gollwitzer und Malzacher [1996] beschreiben ein Experiment, bei dem Versuchspersonen deutlich höhere Erinnerungsleistungen zeigten, wenn sie in Zusammenhang mit vorgenommenen Handlungsgelegenheiten standen.

> Es zeigte sich, daß die in einem Vorsatz spezifizierten Bedingungen für die Ausführung zielrealisierender Handlungen im Gedächtnis besonders leicht zugänglich sind. Dadurch können potentielle Handlungskonflikte (...) durch den Abruf der vorgenommenen Handlungsgelegenheiten und -mittel schnell beendet werden (S.445).

Gelegenheiten können in erweiterten Verhaltensnetzwerken durch geeignete Relevanzbedingungen realisiert werden. Die Relevanzbedingung sorgt dafür, daß ein Ziel nur in der geeigneten Situation relevant ist. Im obigen Beispiel würde das Ziel 'nach Hause gehen' beim Eintreffen der Relevanzbedingung 'beim Bäckerladen' vom dann relevanteren Ziel 'Brot kaufen' unterbrochen werden. Solange die Relevanzbedingung nicht erfüllt ist, hat das Ziel Brot zu kaufen keinen Einfluß auf die Verhaltensauswahl. Einen vergleichbaren Mechanismus stellen Alarms [Norman, 1997] dar. Sogenannte 'opportunity demons' überwachen die aktuelle Situation nach einer Gelegenheit, eine zielerfüllende Aktion auszuführen. Ist eine solche Situation gegeben, wird die Intensität eines Alarms und damit die Wahrscheinlichkeit der Selektion des zugehörigen Ziels erhöht. Dadurch wird wie bei erweiterten Verhaltensnetzwerken gewährleistet, daß der Einfluß von Zielen auf die Verhaltensauswahl auf Situationen beschränkt ist, in denen die Ziele relevant sind.

6.2 Reaktivität

In dynamischen Umgebungen ist eine wichtige Anforderung an einen Handlungskontrollmechanismus, daß er schnell auf Änderungen in der Situation reagiert, um z.B. Gefahren auszuweichen. Erweiterte Verhaltensnetzwerke sind in der Lage, eine schnelle und der momentanen Situation angemessene Verhaltensauswahl zu treffen.

Die Entscheidung zur Ausführung von Verhalten basiert immer auf der aktuellen Situation. Zwar werden durch den Aktivierungsmechanismus des Netzwerks zukünftige Situationen antizipiert und eine Reihenfolge für die Ausführung von Verhalten voraktiviert (siehe auch Kapitel 6.4.2), ändert sich aber die Situation und damit die Ausführbarkeit und Aktivierung von Kompetenzmodulen, kann schnell auf diese Änderung reagiert werden. Wie schnell hängt davon ab, wie lange es dauert, bis die Aktivierung des der neuen Situation angemessenen Verhaltens die höchste Aktivierung aller dieselben Ressourcen benötigenden Verhalten besitzt und die Aktivierungsschwelle übersteigt. Das ist um so schneller der Fall, je direkter ein Verhalten ein Ziel erfüllt und je wichtiger und relevanter dieses Ziel ist. Wichtige Ziele, wie z.B. die Selbsterhaltung, können so beim Eintreten der Relevanzbedingung 'Gefahr' schnell entsprechende (Flucht-) Verhalten aktivieren.

Die Reaktivität kann durch eine Senkung der beiden Parameter β und θ weiter gesteigert werden. Eine niedrige Trägheit der Aktivierung β erlaubt schnellere Verhaltenswechsel, da Kompetenzmodule mit viel Aktivierung diese schneller verlieren können (siehe Kapitel 6.4.4). Durch die Senkung der Aktivierungsschwelle θ kann die Aktivierung eines Verhaltens schneller θ übersteigen und zur Ausführung gelangen. Das hat aber zur Folge, daß die Antizipationsweite

sinkt. In dynamischen Umgebungen sollten also niedrigere Werte von β und θ zu besseren Ergebnissen führen, während in Domänen mit einer geringen Anzahl unvorhergesehener Änderungen der Umgebung hohe Werte zu bevorzugen sind.

Die Zeitkomplexität für die Berechnung eines Zyklus der Aktivierung aller Kompetenzmodule erweiterter Verhaltensnetzwerke beträgt bei serieller Berechnung $O(|\mathcal{M}|^2|\mathcal{G}|)$, wobei $|\mathcal{M}|$ die Anzahl Kompetenzmodule und $|\mathcal{G}|$ die Anzahl Ziele repräsentiert. Jedes Modul kann potentiell mit jedem anderen verbunden sein, und jede Verbindung kann Aktivierung von jedem Ziel führen (Goaltracking). Da erweiterte Verhaltensnetzwerke auch parallel ausgeführt werden können, beträgt die Zeitkomplexität bei der Ausführung auf einer Maschine mit $|\mathcal{M}|$ Prozessoren nur noch $O(|\mathcal{M}||\mathcal{G}|)$

Decugis und Ferber [1998] schlagen eine Variante von Verhaltensnetzwerken mit Zeitkomplexität $O(|\mathcal{M}||\mathcal{S}|)$ (bei serieller Ausführung) vor, wobei $|\mathcal{S}|$ die Anzahl Propositionen in $\mathcal{S}$ angibt. Anstatt Kompetenzmodule direkt miteinander zu verbinden, führen sie Knoten ein, deren Aktivierung den gewünschten Zustand einer Proposition angibt[1]. Die Aktivierungsausbreitung erfolgt dann indirekt über diese Knoten. Der Vorteil dieses Verfahrens ist, daß der Nutzen einzelner Propositionen direkt abgelesen werden kann. Der Nachteil ist, daß in der Regel deutlich mehr Propositionen als Verhaltensmodule existieren.

In der Praxis erweist sich aber auch die serielle Simulation von erweiterten Verhaltensnetzwerken bereits als recht effizient. Im Rahmen des RoboCup99 konnten alle elf Fußballagenten auf einem einzigen PC ausgeführt werden, dessen CPU damit nur zu 50 Prozent ausgelastet war. Andere Mannschaften benötigten bis zu fünf PCs, bzw. bis zu fünf Sun Workstations für ihre elf Agenten. Da jeder Spieler alle 100 ms eine Aktion zum Server schicken kann, bleiben für einen Agenten gerade 9 ms Rechenzeit pro Aktion, von denen nur ein Bruchteil für die Verhaltensauswahl durch das Verhaltensnetzwerk zur Verfügung steht, da auch Wahrnehmung und Verhaltensausführung Zeit in Anspruch nehmen.

6.3 Robustheit

Die situationsangepaßte und schnelle Verhaltenskontrolle macht erweiterte Verhaltensnetzwerke robust gegenüber nicht-deterministischen Effekten, bzw. fehlerhaften Effektoren. Treten Effekte nicht ein, bzw. treten unerwartete Effekte ein, wird dadurch zwar die Qualität der Handlung mit zunehmender Unzuverlässigkeit der Effektoren langsam schlechter, da nicht der direkte Weg zum Ziel verfolgt wird. Anders als bei der Ausführung eines Handlungsplans, wo ein nicht eingetretener Effekt einen Plan nutzlos machen kann, wird die Verhaltensauswahl in erweiterten Verhaltensnetzwerken immer aufgrund der aktuell wahrgenommenen Situation getroffen und damit der neuen, unerwarteten Situation angepaßt.

[1] Tatsächlich werden keine neuen Knoten eingeführt, sondern es werden die bereits vorhandenen Wahrnehmungsknoten doppelt genutzt, was aber letztendlich auf dasselbe hinaus läuft.

Die Robustheit von erweiterten Verhaltensnetzwerken gegenüber fehlerhaften Effektoren wurde in einer empirischen Untersuchung in der RoboCup-Domäne überprüft. Der wichtigste Effektor eines Agenten, die Ballkontrolle, wurde zusätzlich zur vom Server vorgegebenen Ungenauigkeit prozentual verrauscht. Stärke und Richtung eines Schusses wurden also mit zunehmender Ungenauigkeit ausgeführt. Die Ergebnisse von jeweils 30 Spielen je Fehlerstärke einer Mannschaft mit elf Spielern mit fehlerhaften Effektoren gegen eine Mannschaft ohne fehlerhafte Effektoren (d.h. mit der vom Server vorgegebenen Ungenauigkeit) sind in Abbildung 6.3 dargestellt. Die Qualität der verrauschten Mannschaft nimmt mit zunehmender Ungenauigkeit erwartungsgemäß ab, was sich in der sich verschlechternden Tordifferenz zeigt. Wie die Quantität der Abnahme einzuschätzen ist, läßt sich am besten (wenn auch nicht signifikant) durch den Vergleich mit Ergebnissen der RoboCup WM zeigen. Bei 70% zusätzlichem Rauschen beträgt die mittlere Tordifferenz 3,2 Tore. Die beiden Spiele gegen den dritten der Weltmeisterschaft endeten mit einer mittleren Tordifferenz von 4,0 Toren. Trotz 70% Rauschen im wichtigsten Effektor der Fußballagenten ist die Qualitätseinbuße der Handlungskontrolle vergleichsweise gering.

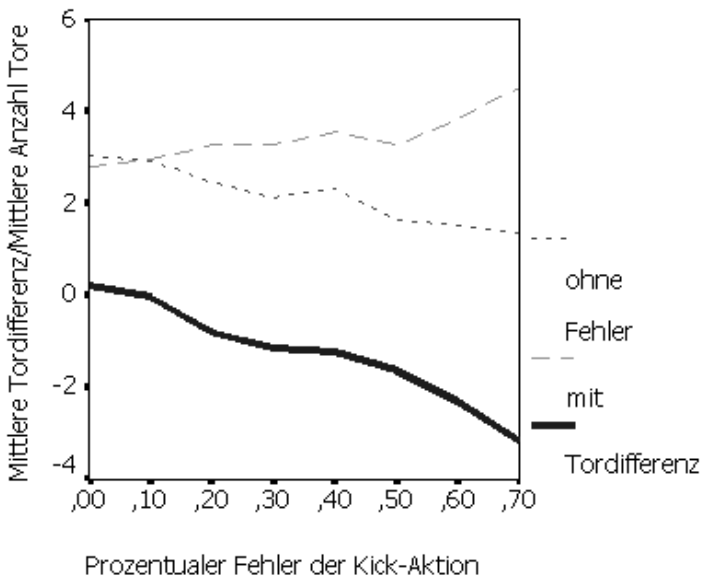

Abbildung 6.3: Qualität einer Mannschaft mit elf Spielern bei zunehmender Verrauschung von Stärke und Richtung der Ball-Kick-Aktion.

Da die Entscheidung für Verhalten in erweiterten Verhaltensnetzwerken aufgrund der aktuell wahrgenommenen Situation getroffen wird, sollten erweiterte Verhaltensnetzwerke tendenziell anfällig gegenüber fehlerhaften Wahrnehmungen sein. Trotzdem sollte die Qualität der Handlungskontrolle nicht abrupt einbrechen, sondern mit zunehmendem Fehler langsam geringer werden (graceful degradation). Dies wurde in einer weiteren empirischen Untersuchung überprüft. Zusätzlich zur vom Server vorgegebenen Ungenauigkeit der Wahrnehmung wurden Richtungs- und Entfernungsangaben aller sichtbaren Objekte

zufällig verrauscht. Der Grad der Verrauschung gibt den prozentualen Fehler der Entfernung und relativen Richtung sichtbarer Objekte an. Er wurde bei einer Mannschaft mit elf Spielern, wie sie im RoboCup99 verwendet wurde, zwischen 0% und 35% variiert. Für jede Variation wurde eine Serie von 30 Spielen gegen eine Mannschaft mit normaler, d.h. nur vom Server verrauschter Wahrnehmung durchgeführt. Die Ergebnisse sind in Abbildung 6.4 dargestellt.

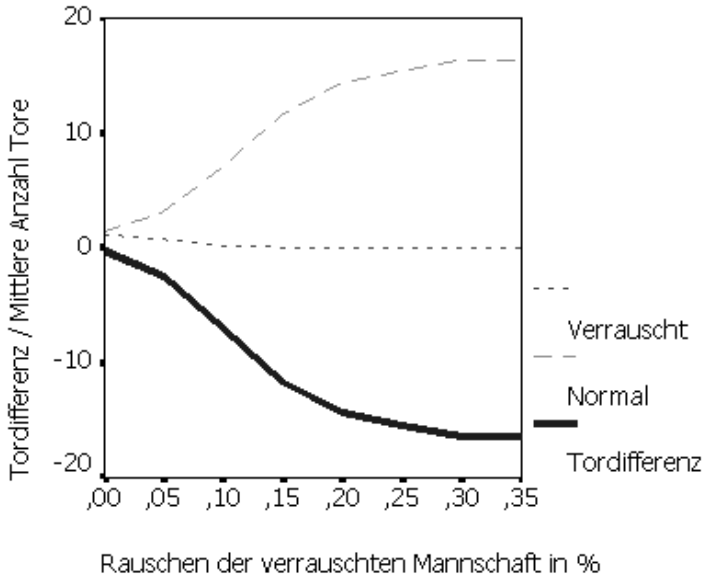

Abbildung 6.4: Qualität einer Mannschaft mit elf Spielern bei zunehmender Verrauschung der Wahrnehmungen.

Die Qualität der verrauschten Mannschaft nimmt mit zunehmender Ungenauigkeit erwartungsgemäß ab. Bei 10% zusätzlichem Rauschen - was das Rauschen des Servers bereits mehr als verdoppelt - liegt die mittlere Tordifferenz bei 7,0 Toren. Das Halbfinale im RoboCup endete im Vergleich dazu mit einem 11:0, Spiele gegen eine weitere Mannschaft unter den besten acht Mannschaften endeten mit 8:0, bzw. 9:0. D.h., noch bei doppelter Ungenauigkeit der Wahrnehmungen hätte das Spielniveau einen Platz unter den besten acht Mannschaften erlaubt. Bei ca. 30% Rauschen tritt ein Bottom-Effekt ein, da die Spielzeit die mögliche Anzahl von Gegentoren begrenzt.

6.4 Einschränkungen und zukünftige Arbeiten

In diesem Kapitel wird erläutert, welchen Einschränkungen die im Rahmen dieser Arbeit vorgestellten erweiterten Verhaltensnetzwerke unterliegen, und wie diese Einschränkungen teilweise überwunden werden können.

6.4.1 Lernen von Verhaltensnetzwerken

Die im Rahmen dieser Arbeit verwendeten Verhaltensnetzwerke, die Ziele, deren Wichtigkeit und Relevanz, die Verhalten und Erfolgswahrscheinlichkeiten wurden manuell erstellt bzw. ermittelt. Einerseits ist es sicherlich positiv, wenn es möglich ist, dem Agenten Wissen auf einfache Weise vorzugeben. Bei neuronalen Netzwerken ist die Vorgabe von Wissen beispielsweise nicht möglich. Diese müssen immer von Grund auf neu lernen. Andererseits wird es mit zunehmender Komplexität der Umgebung immer aufwendiger, alle Effekte und Effektwahrscheinlichkeiten zu ermitteln. Außerdem muß man davon ausgehen, daß sich eine Umgebung im Laufe der Zeit auch ändern kann. Der Agent sollte in der Lage sein, sich solchen Änderungen der Umgebung durch Lernen anzupassen. Ideal wäre, wenn nach der Vorgabe von Vorwissen dieses vom Agenten an die momentane Umgebung adaptiert werden könnte. In einer Phase postaktionaler Erfolgskontrolle, wie sie auch beim Menschen stattfindet [Heckhausen, 1989], könnten die erwarteten Effekte mit den tatsächlich eingetretenen verglichen und, wenn nötig, adaptiert werden.

Lernen von Aktivierungsverbindungen

Maes [1992] beschreibt eine Methode zum Lernen von Effekten von Kompetenzmodulen und damit zum Lernen der Verbindungen eines Verhaltensnetzwerks. Immer wenn sich der Wahrheitswert einer Vorbedingung eines Kompetenzmoduls ändert, wird eine Verbindung zum vorher aktiven Kompetenzmodul erstellt oder entsprechend verstärkt. Die Art der Verbindung hängt davon ab, ob die Vorbedingung falsch wurde (Konfliktorverbindung) oder wahr wurde (Vorgängerverbindung). Besitzt ein Kompetenzmodul eine Verbindung, und wird das zugehörige Modul ausgeführt, ohne daß sich die Vorbedingung ändert, wird das Gewicht der Verbindung entsprechend reduziert. Das Gewicht einer Nachfolgerverbindung eines Kompetenzmoduls wird verstärkt, bzw. eine Nachfolgerverbindung wird angelegt, wenn das Nachfolgermodul direkt nach diesem Modul ausgeführt wurde. Ist das nicht der Fall, wird das Gewicht einer bestehenden Nachfolgerverbindung verringert. Um genügend Exploration zu gewährleisten wird die Verhaltensauswahl so geändert, daß Aktivierungsausbreitung solange stattfindet, bis ein stabiler Aktivierungszustand erreicht ist. Dann wird zufällig eines der ausführbaren Kompetenzmodule ausgeführt, wobei die Wahrscheinlichkeit der Selektion durch die Aktivierung bestimmt wird. Dadurch wird nicht immer das aktivste Modul ausgeführt, sondern es werden auch andere Verhalten ausprobiert.

Diese Methode zum Lernen von Verhaltensnetzwerken besitzt einige Probleme (siehe auch [Maes, 1992]), die die Anwendbarkeit in kontinuierlichen und dynamischen Domänen stark einschränken. So wird davon ausgegangen, daß ein Effekt direkt nach der Ausführung eines Verhaltens eintritt, was in kontinuierlichen Umgebungen in der Regel nicht der Fall ist. Ein Tor fällt beispielsweise erst einige Zeit nach dem Torschuß, abhängig vom Abstand des Agenten vom Tor. Führt der Agent nach dem Torschuß noch weitere Verhalten aus, ist es schwierig, den Effekt 'Tor' dem richtigen Verhalten zuzuweisen. Es wäre not-

wendig, neben der Erwartungswahrscheinlichkeit des Eintreffens eines Effekts auch den erwarteten Zeitpunkt, bzw. Zeitraum zu berücksichtigen.

Weiterhin können Effekte erst nach mehrmaliger Ausführung eines Verhaltens auftreten. Das Verhalten 'zum Ball rennen' wird beispielsweise erst nach mehrmaliger Ausführung, abhängig vom Abstand des Balls, den Effekt 'Ballbesitz' liefern. Jede Ausführung des Verhaltens vor dem Erreichen des Balls würde nach Maes als erfolgloses Verhalten im Bezug auf die Erreichung des Effekts 'Ballbesitz' gewertet werden.

Bei erweiterten Verhaltensnetzwerken kommt das Problem hinzu, daß es bei gleichzeitiger Ausführung von Verhalten schwierig ist, die Änderungen der Umgebung dem jeweils dafür verantwortlichen Verhalten zuzuordnen. Schießt der Agent z.B. den Ball und dreht gleichzeitig den Kopf, ist die Änderung der wahrgenommenen Ballposition eine Folge beider Verhalten.

Besonders in dynamischen Multiagenten-Umgebungen ist schließlich die Annahme fraglich, daß die große Mehrheit der Änderungen der Umgebung durch den Agenten selbst verursacht werden und andere Änderungen in zufälliger Verteilung stattfinden. In der Fußballumgebung sind die Änderungen eines Agenten nur ein kleiner Teil der Änderungen aller 22 Spieler.

Diese Probleme machen es prinzipiell schwierig, Effekte von Verhalten und somit die Aktivierungsverbindungen in Verhaltensnetzwerken in dynamischen und kontinuierlichen Umgebungen zu lernen. Fortschritte im Bereich des Reinforcement Lernens mit verzögerter Verstärkung [Mataric, 1997; Stone, 1998] lassen jedoch hoffen, diese Probleme zu überwinden.

Lernen von Verhalten

Erweiterte Verhaltensnetzwerke stellen einen Mechanismus zur *Auswahl* von Verhalten in einer Situation dar. Die Verhalten selbst werden als gegeben vorausgesetzt, d.h. es wird davon ausgegangen, daß nach der Auswahl von Verhalten die Ausführung dieser ohne weitere Kontrolle erfolgen kann. Wünschenswert wäre, wenn vorhandene Verhalten durch Erfahrung verbessert und neue, komplexere Verhalten aus bestehenden Basisverhalten durch assoziatives Lernen erlangt werden könnten.

Stone und Veloso [1998] beschreiben ein Verfahren (*Layered Learning*), mit dem Verhalten in Stufen gelernt werden. Zunächst werden einfache Verhalten mit wenigen Einflußgrößen durch Erfolg und Irrtum gelernt. Darauf aufbauend werden dann immer komplexere Verhalten aus den einfachen Basisverhalten gebildet. Prinzipiell ähnliche Verfahren sind die *Produktionskompilation* in Act-R [Anderson und Lebiere, 1998] und das *Chunking* in Soar [Rosenbloom *et al.*, 1993], die ebenfalls durch Zusammenfassen von einfachen Produktionen komplexere Produktionen erstellen. Im Zusammenhang mit deklarativem Wissen wird diese Art des Lernens auch als Explanation-Based Learning (EBL) bezeichnet.

In Verhaltensnetzwerken können durch Verstärkung von Nachfolgerverbindungen Verhaltenssequenzen herausgebildet und verstärkt werden. Die Adaption bestehender Verhalten oder das Zusammenfassen mehrerer Verhalten zu einem neuen Verhalten ist aber nicht verwirklicht.

Lernen von Zielen

Unabhängig von der Architektur benötigt der Agent zum Erlernen von Zielen Information über den Nutzen von Zuständen. Gelangt der Agent in einen Zustand mit hohem Nutzen, kann dieser als neues Ziel der Menge der Ziele hinzugefügt werden, wobei die Wichtigkeit des Ziels dem Nutzen des Zustands entspricht. In komplexen Umgebungen wird so die Zahl der Ziele aber rasch wachsen, da in der Regel nur wenige Aspekte (Propositionen) eines Zustands zum Nutzen beitragen. Jede ähnliche Situation wird zur Bildung eines neuen Ziels führen. Gelernt wird also eine Wertefunktion der Zustände der Umgebung. Um die Zahl der Ziele zu reduzieren, ist es notwendig, über die so erlangten Zielzustände zu generalisieren, und nur die relevanten Aspekte als Zielbedingung zu speichern. Besonders wenn Zielzustände nur selten erreicht werden, ist das Lernen und Generalisieren von Zielen nur durch sehr lange Lernphasen zu erreichen [Dorer, 1996]. Die Möglichkeit zur expliziten Vorgabe von Zielen bei erweiterten Verhaltensnetzwerken kann das Erlernen von Zielen wesentlich beschleunigen.

6.4.2 Repräsentation von Zeit

Erweiterte Verhaltensnetzwerke können Zeit nicht explizit repräsentieren. Daher stellt sich die Frage, inwieweit die Reihenfolge, bzw. der Zeitpunkt der Abarbeitung von Zielen und Verhalten in erweiterten Verhaltensnetzwerken gesteuert werden kann.

Ziele

Der Zeitpunkt und somit die Reihenfolge der Abarbeitung von Zielen läßt sich mit Hilfe der Relevanzbedingungen der Ziele kontrollieren. Voraussetzung ist, daß entsprechende Propositionen existieren, deren Wahrheitswert mit dem Termin des Ziels korreliert. Die explizite Repräsentation von Zeit und Zeitdauer kann also in der τ_P Funktion stattfinden. Wächst der Wahrheitswert $\tau_P(a)$ einer solchen Proposition a kontinuierlich mit nahendem Termin, kann so auch der zunehmenden Dringlichkeit des Ziels Rechnung getragen werden. Während ein Ziel weit vor dem Erledigungstermin durch fehlende Relevanz keinen Einfluß auf die Verhaltensauswahl hat, wird mit nahendem Termin der Einfluß des Ziels zunehmend größer und wird, bei entsprechender Wichtigkeit des Ziels, zielerfüllende Verhalten ausreichend aktivieren.

Einen ähnlichen Mechanismus nutzt VMattie [Franklin *et al.*, 1996], um e-Mails termingerecht zu verschicken. Ein Verfahren zur expliziten Repräsentation von Zeit und Zeitdauer stellen Alarms [Norman und Long, 1995] dar. Alarms speichern den gewünschten Zeitpunkt der Zielerreichung sowie die geschätzte Dauer der Bearbeitung und aktivieren zur gegebenen Zeit das Ziel ähnlich den Relevanzbedingungen in erweiterten Verhaltensnetzwerken. Die Ψ-Maschine [Dörner, 1999] realisiert Dringlichkeit von Zielen durch das Hinzufügen von zusätzlichen, das eigentliche Motiv unterstützenden Motiven.

Verhalten

Die Reihenfolge, in der Verhalten ausgeführt werden, ist bei erweiterten Verhaltensnetzwerken durch ihre kausale Beziehung festgelegt. Von den Zielen ausgehend wird Aktivierung an zielerfüllende Verhalten weitergegeben. Sind deren Vorbedingungen nicht erfüllt, geben sie Aktivierung weiter an Verhalten, die diese Vorbedingungen erfüllen können, usw. Dadurch entsteht ein impliziter Plan, durch den die Reihenfolge der Abarbeitung von Verhalten festgelegt ist. Diese Reihenfolge wird eingehalten, solange sich die Umgebung nur im erwarteten Ausmaß ändert.

Die Vorgabe eines Zeitpunkts zur Ausführung eines Verhaltens kann wiederum nur durch die Verwendung einer geeigneten Vorbedingung gewährleistet werden und setzt wie bei den Relevanzbedingungen der Ziele das Vorhandensein einer entsprechenden Proposition voraus. Die Reihenfolge, in der die einzelnen Vorbedingungen eines Kompetenzmoduls erfüllt werden sollen, kann in erweiterten Verhaltensnetzwerken jedoch nicht spezifiziert werden. Der Vorschlag von Rhodes [1996], bei Verhalten mit einer festgelegten Reihenfolge von Vorbedingungen Aktivierung nur von der ersten nicht erfüllten Vorbedingung weiterzugeben, ist in kontinuierlichen Umgebungen nur bedingt anwendbar, da der Begriff der ersten nicht erfüllten Vorbedingung unscharf ist.

Lalanda und Hayes-Roth [1994] weisen mit Hilfe eines Zeitplans (control schedule) einzelnen zielerfüllenden Aufgaben einen Zeitstempel zur Bearbeitung zu. In Guardian [Hayes–Roth, 1995], einem System zur Überwachung von Patienten auf Intensivstationen, werden die entsprechenden Auslöser von Verhalten Trigger genannt, mit deren Hilfe die zeitliche Koordination der einzelnen Verhalten erfolgt.

6.4.3 Variablen

Verhaltensnetzwerke sind nicht in der Lage, Variablen zu repräsentieren. Verhalten wie z.B. 'stack(X, Y)' in der Blockwelt sind nicht spezifizierbar. Trotzdem wurden Anstrengungen unternommen, Variablen in Verhaltensnetzwerken zuzulassen.

Die einfachste Möglichkeit, die im Rahmen dieser Arbeit bei erweiterten Verhaltensnetzwerken für die Blockwelt eingesetzt wurde, ist die vollständige Instanziierung aller Variablen bei der Erstellung des Netzwerks. Während eine vom Benutzer spezifizierte Verhaltensregel Variablen enthalten darf, wird bei der Erstellung des Netzwerks für jede mögliche Objekt-Instanziierung ein neues Kompetenzmodul angelegt (Bsp. stack(a, b), stack(a, c), ...). Dies ist jedoch nur bei einer endlichen Anzahl im voraus bekannter Objekte möglich und nur bei einer relativ geringen Anzahl Objekte sinnvoll, da die Zahl der Kompetenzmodule exponentiell mit der Zahl vorhandener Objekte steigt.

Ein flexiblerer Ansatz wurde in VMattie [Franklin *et al.*, 1996] verfolgt. Dort werden die Verhaltensnetzwerke zur Laufzeit nach Bedarf erzeugt und die Variablen dabei mit den konkret vorliegenden Objekten instanziiert. Dadurch wird die bei vollständiger Instanziierung potentiell exponentielle Aufblähung der Anzahl von Kompetenzmodulen vermieden. Ein Netzwerk wird wieder gelöscht,

wenn das Ziel erreicht wurde. Unklar bleibt jedoch, wie bei Netzwerken mit mehreren Zielen, bzw. bei graduellen Zielen in kontinuierlichen Umgebungen vorgegangen wird.

Ein Verfahren zur klassischen Weitergabe von Variablen in Verhaltensnetzwerken wird in [Rhodes, 1996] beschrieben. Zusammen mit der Aktivierung werden über Nachfolger- und Konfliktorverbindungen Objekte an Kompetenzmodule weitergegeben, mit denen die dort vorhandenen Variablen instanziiert werden. Filter sorgen dafür, daß nur Objekte des richtigen Typs weitergegeben werden. Das Verfahren ist jedoch auf die Weitergabe lediglich einer Variable begrenzt.

Maes selbst schlägt vor [1989], index-funktionale Objekte [Agre und Chapman, 1987] anstelle von Variablen zu verwenden. Anstatt Objekte absolut zu benennen, werden sie bezüglich ihrer Funktion in der momentanen Situation indiziert. Dadurch kann die Zahl der zu repräsentierenden Objekte deutlich verringert werden, wodurch die Verwendung von Variablen in vielen Domänen überflüssig wird. Dieser Ansatz wurde bei allen Verhaltensnetzwerken der RoboCup-Domäne erfolgreich eingesetzt (siehe auch Kapitel 4.4).

6.4.4 Persistenz des Verhaltens

Atkinson und Birch [1970] weisen auf ein Problem hin, das entsteht, wenn die Reduktion einer Handlungstendenz davon abhängt, ob die Handlung ausgeführt wird oder nicht. Wenn eine Handlungstendenz so groß wird, daß die Handlung ausgeführt wird, wird die Tendenz dadurch wieder niedriger und von der zuvor dominanten Handlungstendenz übertroffen. Dadurch können Handlungsoszillationen (behavioral chatter [Kuhl, 1982]) entstehen.

In ähnlicher Weise können bei erweiterten Verhaltensnetzwerken bei der Verfolgung von Zielen mit kontinuierlicher Relevanz Handlungsoszillationen auftreten. Ein Beispiel aus der Fußballumgebung ist in (Abb. 6.5) dargestellt. Das Verhalten 'zum Ball rennen' benötigt Ausdauer. Je weniger Ausdauer der Agent besitzt, desto relevanter wird das Ziel, ausgeruht zu sein. Hat der Spieler sich kurz ausgeruht, sinkt die Relevanz des Ziels 'ausgeruht sein' und die Aktivierung des Verhaltens 'ausruhen' sinkt wieder unter die Aktivierung des Verhaltens 'zum Ball rennen'. Eine Handlungsoszillation entsteht.

Abbildung 6.5: Handlungsoszillation

Erweiterte Verhaltensnetzwerke bieten mit der Trägheit der Aktivierung - gesteuert durch den Parameter β - eine Möglichkeit, Handlungsoszillationen zu dämpfen. Tatsächlich zeigt sich in Versuchen in der Fußballumgebung, daß bei einer Erhöhung von β die Zahl der Verhaltenswechsel sinkt, und somit stabileres Verhalten erreicht wird (Abb. 6.6). Trotzdem ist dieses Ergebnis unbefriedigend,

da mit der erhöhten Trägheit des Verhaltens auch dessen Qualität in der Fußballdomäne nachläßt. Die höhere Trägheit aller Verhalten erweist sich als zu unspezifisch.

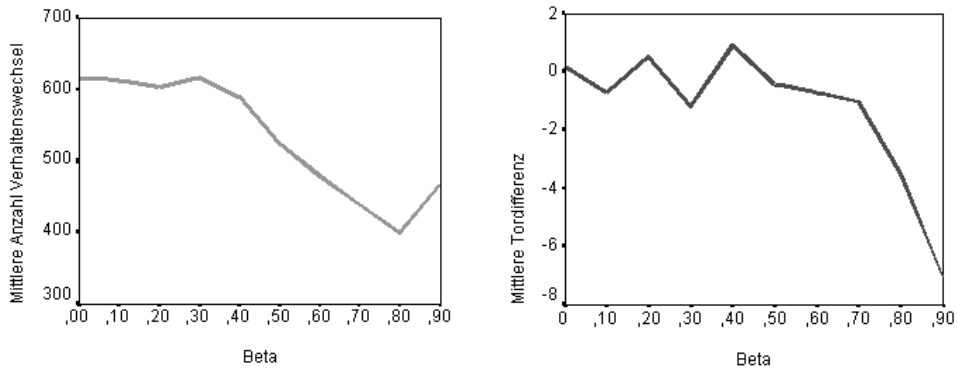

Abbildung 6.6: Verhaltenswechsel (links) und Tordifferenz (rechts) in Abhängigkeit der Aktivierungsträgheit β.

Eine spezifischere Alternative zum Vermeiden von Handlungsoszillationen in erweiterten Verhaltensnetzwerken wäre die Aktivierung von Zielen durch konsummatorische, d.h. direkt zielerfüllende Verhalten, wie sie in der Motivationspsychologie vorgeschlagen wird: "Nicht nur die äußere und innere Reizsituation, sondern auch die Handlungsausführung selbst bewirken eine Veränderung der Motivation" [Becker–Carus, 1983]. Die Auswahl eines konsummatorischen Verhaltens könnte eine Aktivierungsrückkopplung mit dem zugehörigen Ziel initiieren, die das in Gang gesetzte Verhalten gegenüber anderen Verhalten abschirmt. Zu klären wäre dann, wie und wann ein solches selbsterhaltendes Verhalten unterbrochen, bzw. beendet werden kann. Eine Möglichkeit bestünde darin, die Aktivierung eines Ziels durch ein konsummatorisches Verhalten als Änderung der τ_P-Funktion der Relevanzbedingung des Ziels aufzufassen (siehe Abb. 6.7). Die Relevanz des Ziels würde bei Ausführung eines konsummatorischen Verhaltens langsamer abfallen als sie vor der Ausführung bei steigender Dringlichkeit angestiegen ist.

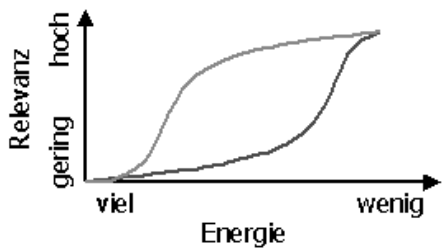

Abbildung 6.7: Unterschiedliche Relevanzbedingung vor der Ausführung (unten) und während der Ausführung (oben) eines konsummatorischen Verhaltens zur Vermeidung von Handlungsoszillationen.

Kuhl [1982] schlägt vor, Handlungsoszillationen durch selektive Aufmerksamkeit zu vermeiden. Eine metakognitive Handlungstendenz soll bei sich annäherndem Unterschied zweier Handlungstendenzen aktiv werden und die vorherrschende Handlungstendenz verstärken. Dadurch wird erst bei deutlicher Überschreitung der vorherrschenden Handlungstendenz eine neue Handlung selektiert. Handlungsoszillationen werden so aber nicht vermieden, es wird lediglich die Frequenz der Oszillation erniedrigt. Dörner [1996] führt zur Vermeidung von Handlungsoszillationen laterale Inhibition der herrschenden Absicht gegenüber konkurrierenden Absichten ein. Eine einmal ausgewählte Absicht kann so andere Absichten hemmen, wodurch ein stabileres Verhalten erzielt wird. Wie die Trägheit der Aktivierung in Verhaltensnetzwerken kann aber auch dieser Mechanismus zu unspezifisch sein und wird in dynamischen Umgebungen zu verminderter Reaktivität führen.

Kapitel 7

Schlußbemerkung

Wie nicht zuletzt das Erreichen des Vizeweltmeistertitels beim RoboCup99 gezeigt hat, sind erweiterte Verhaltensnetzwerke zur erfolgreichen Handlungskontrolle autonomer Agenten in der Lage. Besonders wichtig ist es jedoch, sich klar zu machen, in welchen Domänen erweiterte Verhaltensnetzwerke sinnvoll eingesetzt werden können. Zwar können erweiterte Verhaltensnetzwerke auch in statischen, diskreten und deterministischen Umgebungen wie der Blockwelt eingesetzt werden. Sie sind dort aber z.B. Handlungsplanern unterlegen, da diese Vollständigkeit und meist auch Optimalität einer Lösung garantieren. In dynamischen, kontinuierlichen und nicht-deterministischen Umgebungen ist jedoch die Berechnungskomplexität solcher Verfahren meist prohibitiv. Erweiterte Verhaltensnetzwerke können in solchen Umgebungen schnell und zielgerichtet agieren und sind damit sowohl rein deliberativer Handlungskontrolle als auch rein reaktiver Verhaltenskontrolle überlegen. Weitere Eigenschaften, wie die statische und dynamische Priorisierbarkeit von Zielen, die Möglichkeit, Verhalten gleichzeitig auszuführen, sowie die Parallelisierbarkeit der Berechnung heben erweiterte Verhaltensnetzwerke zudem von anderen Handlungskontrollmechanismen ab (Tab. 7.1).

	Korrekt	Vollständig	Optimal	Dynamische Umgebung	Nicht-deterministische U.	Teilweise zugängliche U.	Kontinuierliche Umgebung	Ziele priorisierbar	Gleichzeitige Verhalten	Berechnung parallelisierbar	Adaptiv
EBN	√	-	-	√	√	√	√	√	√	√	(√)
MASM	√	-	-	√	√	√	-	-	-	(√)	(√)
STRIPS	√	√	√	-	-	-	-	-	-	-	-
POMDPs	√	√	√	√	√	√	-	-	(√)	-	√
Act-R	√	-	-	(-)	√	√	(√)	√	-	(√)	√
Soar	√	-	-	√	√	√	(√)	-	-	(√)	√

Tabelle 7.1: Eigenschaften und Anwendungsbereich verschiedener Handlungskontrollmechanismen.

Anhang A

Detailierte statistische Ergebnisse

In diesem Kapitel sind die Einzelergebnisse statistischer Untersuchungen widergegeben, die in Kapitel 5 zur besseren Übersicht als Diagramme dargestellt wurden.

A.1 Optimierung der Parameter

gamma	variierte Mannschaft	statische Mannschaft	Tordifferenz
0.0	0.00	18.00	-18.00
0.1	6.97	6.83	0.14
0.2	8.07	7.00	1.07
0.3	7.65	6.74	0.91
0.4	8.27	6.83	1.44
0.5	7.23	7.13	0.10
0.6	7.00	6.29	0.71
0.7	5.23	6.83	-1.60
0.8	5.63	7.27	-1.64
0.9	5.77	7.67	-1.90

Tabelle A.1: Ergebnisse der in Abb. 5.1 dargestellten Parametervariation des Aktivierungsparameters γ.

A.2 Verhaltensparametrisierung

statischer Verhaltensparameter	Tore statisch	Tore dynamisch	p	n
0.0	2.47	22.40	< 0.001	30
0.1	3.87	20.23	< 0.001	30
0.2	4.50	18.83	< 0.001	30
0.3	5.80	17.30	< 0.001	30
0.4	7.10	15.00	< 0.001	30
0.5	8.83	14.30	< 0.001	30
0.6	8.02	11.91	< 0.001	45
0.7	8.89	11.20	0.003	45
0.8	9.38	11.31	0.005	45
0.9	9.00	11.60	0.001	45
1.0	7.51	11.00	< 0.001	45

Tabelle A.2: Signifikanz der in Abb. 5.3 dargestellten Ergebnisse des Vergleichs von statischer und dynamischer Verhaltensparametrisierung.

statischer Verhaltensparameter	Pausen statisch	Pausen dynamisch	p	n
0.0	72.5	113.8	< 0.001	30
0.1	84.6	121.2	< 0.001	30
0.2	84.4	114.7	< 0.001	30
0.3	93.9	118.1	< 0.001	30
0.4	94.2	116.5	< 0.001	30
0.5	107.7	132.7	< 0.001	30
0.6	122.5	132.2	0.021	45
0.7	130.6	130.2	0.95	45
0.8	142.4	124.2	0.002	45
0.9	165.7	136.5	< 0.001	45
1.0	163.0	132.7	< 0.001	45

Tabelle A.3: Ergebnisse des in Abb. 5.4 dargestellten Vergleichs der mittleren Anzahl Erholungspausen bei statischer und dynamischer Parametrisierung.

A.3 Reelwertige Propositionen

Rundungsgrenze	Tore binär	Tore reellwertig	p ($n = 60$)
0.1	8.29	7.25	0.45
0.2	8.23	8.82	0.18
0.3	7.75	8.33	0.27
0.4	7.40	8.33	0.05
0.5	6.57	8.03	0.006
0.6	5.92	7.45	0.002
0.7	5.62	7.42	< 0.001
0.8	5.58	7.07	0.002
0.9	5.08	7.32	< 0.001

Tabelle A.4: Signifikanz der in Abb. 5.5 dargestellten Ergebnisse des Vergleichs von binären und reellwertigen Propositionen.

Rundungsgrenze	Tore binär	Tore reellwertig	p ($n = 30$)
0.1	6.53	9.13	< 0.001
0.2	4.43	8.27	< 0.001
0.3	6.53	8.57	0.021
0.4	5.60	6.77	0.085
0.5	4.23	7.40	< 0.001
0.6	3.33	8.03	< 0.001
0.7	2.47	9.70	< 0.001
0.8	2.27	10.13	< 0.001
0.9	1.27	8.97	< 0.001

Tabelle A.5: Signifikanz der in Abb. 5.6 dargestellten Ergebnisse des Vergleichs von binären und reellwertigen Propositionen in Netzwerken mit vier Zielen.

Rundungsgrenze	Tore binär	Tore reellwertig	p ($n = 30$)
0.1	0.70	4.90	< 0.001
0.2	0.69	5.73	< 0.001
0.3	0.77	5.53	< 0.001
0.4	0.63	5.43	< 0.001
0.5	0.87	6.10	< 0.001
0.6	0.80	6.10	< 0.001
0.7	0.67	7.00	< 0.001
0.8	0.63	6.70	< 0.001
0.9	0.53	7.43	< 0.001

Tabelle A.6: Signifikanz der in Abb. 5.7 dargestellten Ergebnisse des Vergleichs von binären und reellwertigen Propositionen bei Spielen mit 11 Spielern.

Anhang B

Verhaltensnetzwerke

B.1 Spiele mit zwei Spielern

Bei Spielen mit zwei Spielern besaß jeder Agent dasselbe Netzwerk, ein Rollenverhalten wurde nicht implementiert. Die Ziele sind durch das Schlüsselwort 'Goal' gekennzeichnet und enthalten als Bedingungsteil die Relevanzbedingung und im Konklusionsteil die Zielbedingung und die Wichtigkeit des Ziels. Die Verhaltensregeln enthalten im Bedingungsteil die Vorbedingung des Verhaltens, im Aktionsteil das eigentliche Verhalten und, durch das Schlüsselwort 'Effect' getrennt, die Effekte und deren Eintretenswahrscheinlichkeit.

```
Goal
if ballIsNearOtherGoal
then soccergoal 0.9
endgoal

Goal
if ballIsNearOwnGoal
then noOtherGoal 0.8
endgoal

Goal
if teammateHasBall
then standFree 0.5
endgoal

Goal
if lowStamina
then not lowStamina 0.6
endgoal

If lowStamina
then relax
effect not lowStamina 0.9
endif

If ballKickable
and nearGoal
```

```
then kickGoal
effect not haveBall 0.9
and not ballKickable 0.9
and soccergoal 0.9
endif

If ballKickable
then runWithBall
effect nearGoal 0.6
and soccergoal 0.2
and noOtherGoal 0.1
and not nearOwnGoal 0.4
and lowStamina 0.4
endif

If ballKickable
and teammateIsFree
and teammateIsNearerOtherGoal
then passBall
effect not haveBall 0.9
and not ballKickable 0.9
and soccergoal 0.5
and noOtherGoal 0.2
endif

if ballKickable
and nearOwnGoal
then kickAwayBall
effect not haveBall 0.9
and not ballKickable 0.9
and noOtherGoal 0.8
endif

if not haveSeenBall
then searchBall
effect haveSeenBall 0.5
endif

if haveSeenBall
then runToBall
effect haveBall 0.8
and ballKickable 0.7
and lowStamina 0.4
endif

if opponentIsNearerToBall
and opponentIsNearerToOwnGoal
then runToOwnGoal
effect noOtherGoal 0.5
and not nearGoal 0.6
and lowStamina 0.4
endif
```

```
if teammateIsNearerBall
then runFree
effect standFree 0.6
and nearGoal 0.7
and not nearOwnGoal 0.5
and lowStamina 0.5
endif
```

B.2 RoboCup99

Beim RoboCup99 wurden vier Kategorien von Spielern mit unterschiedlichen Verhaltensnetzwerken eingesetzt: Torwart, Abwehrspieler, Mittelfeldspieler und Stürmer. Sie unterscheiden sich sowohl in der Anzahl und Art der verwendeten Verhalten, als auch in der Priorisierung der Ziele.

Die in der neuen Server Version (5) neu eingeführte Möglichkeit der parallelen Ausführung von Verhalten erlaubte in diesem Zusammenhang erstmals den Test paralleler Verhaltensauswahl in erweiterten Verhaltensnetzwerken. Die verwendeten Ressourcen von Verhalten sind in den Verhaltensregeln durch das Schlüsselwort 'Using' gekennzeichnet. Die Ressourcenknoten werden automatisch anhand der verwendeten Ressourcen erstellt. Im folgenden sind die unterschiedlichen Netzwerke aufgeführt.

B.2.1 Torwart

```
Goal
if ballIsNearOtherGoal
then soccergoal 0.1
endgoal

Goal
if
then noOtherGoal 1.0
endgoal

Goal
if
then haveBall 0.7
endgoal

Goal
if
then teammateHasBall 0.8
endgoal

Goal
if
then haveSeenBall 0.7
endgoal

Goal
```

```
if lowStamina
then not lowStamina 0.5
endgoal

If catchSuccess
then moveToHomePosition
effect haveBall
and noOtherGoal
endif

If ballKickable
and nearGoal
then kickGoal
effect not haveBall 0.9
and not ballKickable 0.9
and soccergoal 0.5
using leg 1
endif

if ballKickable
then kickAwayBall
effect not haveBall 0.9
and not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if not haveSeenBall
then searchBall
effect haveSeenBall 0.5
using leg 1
endif

if haveSeenBall
and not ballKickable
and ballIsVeryNearOwnGoal
and not teammateIsBetweenBall
then runToBall
effect haveBall 0.5
and ballKickable 0.4
using leg 1
endif

if not ballIsVeryNearOwnGoal
then runToHomePosition
effect not lowStamina 0.3
and noOtherGoal 0.5
using leg 1
endif

if teammateIsNearerBall
then runToHomePosition
effect not lowStamina 0.3
```

```
and noOtherGoal 0.5
using leg 1
endif

If ballKickable
and teammateIsFree
then passBall
effect not haveBall 0.9
and not ballKickable 0.9
and teammateHasBall 0.8
using leg 1
endif

If
then relax
effect not lowStamina 0.5
using leg 1
endif

If maySaySomething
then sayPosition
effect teammateHasBall 0.8
using mouth 1
endif

If haveSeenBall
then mindBall
effect haveSeenBall 0.8
using neck 1
endif
```

B.2.2 Abwehrspieler

```
Goal
if ballIsNearOtherGoal
then soccergoal 0.9
endgoal

Goal
if ballIsNearOwnGoal
noOtherGoal 1.0
endgoal

Goal
if ballFarAhead
and not lowStamina
then offsideTrap 0.8
endgoal

Goal
if
then haveBall 0.7
endgoal
```

```
Goal
if
then teammateHasBall 0.8
endgoal

Goal
if lowStamina
then not lowStamina 0.6
endgoal

if ballKickable
and nearOpponent
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if ballKickable
and lowStamina
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if ballKickable
and nearOwnGoal
then kickAwayBall
effect not ballKickable 0.9
and noOtherGoal 0.5
using leg 1
endif

if not haveSeenBall
then searchBall
effect haveSeenBall 0.5
using leg 1
endif

if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
and ballOutside
then turnAroundBallForKickIn
effect haveBall 0.7
and ballKickable 0.4
and lowStamina 0.3
using leg 1
endif
```

```
if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
then runToBall
effect haveBall 0.5
and ballKickable 0.4
and lowStamina 0.3
using leg 1
endif

If ballKickable
and nearGoal
then kickGoal
effect not haveBall 0.9
and not ballKickable 0.9
and soccergoal 0.5
using leg 1
endif

If teammateIsNearerBall
and not teammateBehind
then pushForward
effect offsideTrap 0.3
and lowStamina 0.1
using leg 1
endif

if ballBehind
and teammateIsNearerBall
and not lowStamina
then pushBackward
effect noOtherGoal 0.3
and lowStamina 0.1
using leg 1
endif

If ballKickable
and teammateIsFree
and not freeTeammateIsOffside
then passBall
effect not haveBall 0.9
and not ballKickable 0.9
and teammateHasBall 0.8
using leg 1
endif

if not ballFarAhead
and not ballBehind
and teammateIsNearerBall
then searchBall
effect haveSeenBall 0.5
using leg 1
```

```
endif

If ballKickable
and not nearOwnGoal
and not lowStamina
and not opponentInFront
then runWithBall
effect nearGoal 0.4
and soccergoal 0.1
and noOtherGoal 0.2
and lowStamina 0.1
using leg 1
endif

If
then relax
effect not lowStamina 0.3
using leg 1
endif

If maySaySomething
then sayPosition
effect teammateHasBall 0.8
using mouth 1
endif

If haveSeenBall
then mindBall
effect haveSeenBall 0.8
using neck 1
endif
```

B.2.3 Mittelfeldspieler

```
Goal
if ballIsNearOtherGoal
then soccergoal 1.0
endgoal

Goal
if ballIsNearOwnGoal
then noOtherGoal 0.9
endgoal

Goal
if ballFarAhead
and not lowStamina
then offsideTrap 0.8
endgoal

Goal
if
then haveBall 0.7
```

```
endgoal

Goal
if
then teammateHasBall 0.8
endgoal

Goal
if lowStamina
then not lowStamina 0.6
endgoal

if ballKickable
and nearOpponent
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if ballKickable
and lowStamina
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if ballKickable
and nearOwnGoal
then kickAwayBall
effect not ballKickable 0.9
and noOtherGoal 0.5
using leg 1
endif

if not haveSeenBall
then searchBall
effect haveSeenBall 0.5
using leg 1
endif

if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
and ballOutside
then turnAroundBallForKickIn
effect haveBall 0.7
and ballKickable 0.4
and lowStamina 0.3
using leg 1
endif
```

```
if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
then runToBall
effect haveBall 0.5
and ballKickable 0.4
and lowStamina 0.3
using leg 1
endif

If ballKickable
and nearGoal
then kickGoal
effect not haveBall 0.9
and not ballKickable 0.9
and soccergoal 0.5
using leg 1
endif

If teammateIsNearerBall
then pushForward
effect offsideTrap 0.3
and lowStamina 0.1
using leg 1
endif

if ballBehind
and teammateIsNearerBall
and not lowStamina
then pushBackward
effect noOtherGoal 0.3
and lowStamina 0.1
using leg 1
endif

If ballKickable
and teammateIsFree
and not freeTeammateIsOffside
then passBall
effect not haveBall 0.9
and not ballKickable 0.9
and teammateHasBall 0.8
using leg 1
endif

if not ballFarAhead
and not ballBehind
and teammateIsNearerBall
then searchBall
effect haveSeenBall 0.5
using leg 1
```

```
endif

If ballKickable
and not nearOwnGoal
and not lowStamina
and not opponentInFront
then runWithBall
effect nearGoal 0.4
and soccergoal 0.1
and noOtherGoal 0.2
and lowStamina 0.1
using leg 1
endif

If
then relax
effect not lowStamina 0.3
using leg 1
endif

If maySaySomething
then sayPosition
effect teammateHasBall 0.8
using mouth 1
endif

If haveSeenBall
then mindBall
effect haveSeenBall 0.8
using neck 1
endif
```

B.2.4 Stürmer

```
Goal
if ballIsNearOtherGoal
then soccergoal 1.0
endgoal

Goal
if ballIsNearOwnGoal
then noOtherGoal 0.9
endgoal

Goal
if
then haveBall 0.7
endgoal

Goal
if
then teammateHasBall 0.8
endgoal
```

```
Goal
if lowStamina
then not lowStamina 0.6
endgoal

Goal
if inOtherHalf
then not inOffside 0.6
endgoal

If ballKickable
and nearGoal
then kickGoal
effect not ballKickable 0.9
and soccergoal 0.7
using leg 1
endif

if not haveSeenBall
then searchBall
effect haveSeenBall 0.9
using leg 1
endif

if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
and ballOutside
then turnAroundBallForKickIn
effect haveBall 0.7
and ballKickable 0.4
and lowStamina 0.3
using leg 1
endif

if playmodeGoalkickOther
then awaitGoalkick
effect haveBall 0.5
and ballKickable 0.4
and lowStamina 0.1
using leg 1
endif

if haveSeenBall
and not ballKickable
and not teammateIsNearerBall
and haveEnoughStamina
then runToBall
effect haveBall 0.5
and ballKickable 0.4
and lowStamina 0.3
```

```
using leg 1
endif

if ballKickable
and nearOpponent
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
using leg 1
endif

if ballKickable
and lowStamina
then kickAwayBall
effect not ballKickable 0.9
and teammateHasBall 0.5
and soccergoal 0.1
using leg 1
endif

If ballKickable
and teammateIsFree
and not freeTeammateIsOffside
then passBall
effect not haveBall 0.9
and not ballKickable 0.9
and teammateHasBall 0.8
using leg 1
endif

If ballAhead
and not haveBall
and not lowStamina
then runForwardToBall
effect haveBall 0.3
and nearGoal 0.5
and lowStamina 0.1
using leg 1
endif

If haveBall
and not lowStamina
then runWithBall
effect nearGoal 0.6
and soccergoal 0.1
and noOtherGoal 0.2
and lowStamina 0.1
using leg 1
endif

If inOffside
and not haveBall
and highStamina
```

```
and not teammateInFront
then retreat
effect not inOffside 0.8
and lowStamina 0.1
using leg 1
endif

If ballIsVeryNearOwnGoal
and not haveBall
and not lowStamina
then retreat
effect not inOffside 0.8
and lowStamina 0.1
using leg 1
endif

If inOtherHalf
and not knowNotInOffside
and not inOffside
and not haveBall
then checkOffside
effect not inOffside 0.5
using leg 1
endif

If
then relax
effect not lowStamina 0.5
using leg 1
endif

If maySaySomething
then sayPosition
effect teammateHasBall 0.8
using mouth 1
endif

If haveSeenBall
then mindBall
effect haveSeenBall 0.8
using neck 1
endif
```

B.3 Blockwelt

Die Besonderheit der Blockwelt Verhaltensregeln ist, daß sie Variablen enthalten können. Diese sind als Großbuchstaben gekennzeichnet. Zusätzlich muß die Menge der Instanzen (objects) angegeben werden. Bei der Umsetzung der Netzwerkspezifikation in ein Verhaltensnetzwerk wird dann eine vollständige Instanziierung durchgeführt. Jede Verhaltensregel ist also in jeder Kombination von Variableninstanziierungen als Kompetenzmodul im Netzwerk vorhanden. Wei-

terhin enthalten die Effekte keine Eintretenswahrscheinlichkeit, da die Domäne deterministisch ist. Auch die Wichtigkeit aller Ziel ist identisch (1.0).

```
objects
a
b
c
d
endObjects

attributes
handfree
free(X)
holding(X)
on(X,table)
on(X,Y)
endAttributes

goals
on-d-table
if on-d-table then on-c-d
if on-c-d and on-d-table then on-b-c
if on-b-c and on-c-d and on-d-table then on-a-b
endGoals

initial
on-d-table
on-c-table
on-b-c
on-a-table
free-a
free-b
free-d
handfree
endInitial

if
holding(X)
then putdown(X)
effect
on(X,table)
and free(X)
and handfree
and not holding(X)
endif

if
on(X,table)
and free(X)
and handfree
then pickup(X)
effect
holding(X)
```

```
and not on(X,table)
and not free(X)
and not handfree
endif

if
holding(X)
and free(Y)
then stack(X,Y)
effect
on(X,Y)
and free(X)
and handfree
and not free(Y)
and not holding(X)
endif

if
on(X,Y)
and free(X)
and handfree
then unstack(X,Y)
effect
holding(X)
and free(Y)
and not free(X)
and not handfree
and not on(X,Y)
endif
```

Literaturverzeichnis

[Agre und Chapman, 1987] Ph. E. Agre und D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the AAAI–87 conference*, Seiten 268–272. Morgan Kaufmann, 1987.

[Agre und Chapman, 1990] Ph. E. Agre und D. Chapman. What are plans for. In P. Maes (Ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, Seiten 17–34. Elsevier, Amsterdam, 1990.

[Allais, 1953] M. Allais. Le comportement de l'homme rationel devant le risque: Critique des postulats et axiomes de l'ecole americaine. *Econometrica*, 21, Seiten 503–546, 1953.

[Anderson, 1993] J. R. Anderson. *Rules of the Mind.* Erlbaum, Hillsdale, 1993.

[Anderson und Lebiere, 1998] J. R. Anderson und Ch. Lebiere. *The Atomic Components of Thought.* Erlbaum, Mahwah, NJ, 1998.

[Atkinson, 1957] J. W. Atkinson. Motivational determinants of risk–taking behavior. *Psychological Review*, 64, Seiten 359–372, 1957.

[Atkinson und Birch, 1970] J. W. Atkinson und D. A. Birch. *A dynamic theory of action.* Wiley, New York, 1970.

[Becker–Carus, 1983] Ch. Becker–Carus. Motivationale Grundlagen der Nahrungs– und Flüssigkeitsaufnahme. In H. Thomae (Hrsg.), *Enzyklopädie der Psychologie*, 4(2). Hogrefe–Verlag, Göttingen, 1983.

[Bellman, 1957] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[Blum und Furst, 1997] A. L. Blum und M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2), Seiten 279–298, 1997.

[Bonasso *et al.*, 1996] R. P. Bonasso, D. Kortenkamp, D. P. Miller, und M. Slack. Experiences with an architecture for intelligent, reactive agents. In M. Wooldridge, J. P. Müller, und M. Tambe (Eds.), *Intelligent Agents II (ATAL95)*, Seiten 187–202. Springer, Berlin, 1996.

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, und S. Hanks. Decision–theoretic planning: Structural assumptions and computational leverage. In *Journal of Artificial Intelligence Research*, 11, Seiten 1–94. Morgan Kaufmann, 1999.

[Boutilier *et al.*, 1995] C. Boutilier, R. Dearden, und M. Goldszmidt. Exploiting structure in policy construction. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI–95)*, Seiten 1104–1111, 1995.

[Boutilier und Poole, 1996] C. Boutilier und D. Poole. Computing optimal policies for partially observable decision processes using compact representations. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Seiten 1168–1175, 1996.

[Bratman, 1987] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard Univ. Press, Cambridge, Mass., 1987.

[Bratman, 1990] M. E. Bratman. What is intention ? In P. R. Cohen, J. Morgan, und M. E. Pollack (Eds.), *Intentions in communication*, Seiten 15–31. MIT Press, Cambridge, Mass, 1990.

[Brooks, 1986] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA–2(1), Seiten 14–23, 1986.

[Brooks, 1989] R. Brooks. A robot that walks: Emergent behaviors from a carefully evolved network. In *MIT AI Lab Memo*, 1091. MIT Press, 1989.

[Brooks, 1991] R. Brooks. Intelligence without representation. *Artificial Intelligence*, 47, Seiten 139–160, 1991.

[Burgard *et al.*, 1998] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, und S. Thrun. The interactive museum tour–guide robot. In *Proceedings of the AAAI–98 conference*, Madison, 1998. Morgan Kaufmann.

[Burkhard *et al.*, 1998] H. D. Burkhard, M. Hannebauer, und J. Wendler. Beliefe–Desire–Intention deliberation in artificial soccer. *AI Magazine*, 3, Seiten 87–93, 1998.

[Bylander, 1994] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2), Seiten 165–204, 1994.

[Cassandra *et al.*, 1994] A. R. Cassandra, L. P. Kaelbling, und M. L. Littman. Acting optimally in partially observable stochastic domains. *Proceedings of the AAAI–94 conference*, 1994.

[Chapman, 1987] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3), Seiten 333–377, 1987.

[Chapman, 1989] D. Chapman. Penguins can make cake. In *AI–Magazine*, 10(4), Seiten 45–50. AAAI Press, 1989.

[Cohen und Levesque, 1987] P. R. Cohen und H. J. Levesque. Persistence, intention and commitment. In M. P. Georgeff und A. L. Lansky (Eds.), *Proceedings of the 1986 workshop on Reasoning about Actions and Plans*, Seiten 297–340. Morgan Kaufmann Publishers, San Mateo, CA, 1987.

[Decugis und Ferber, 1998] V. Decugis und J. Ferber. Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In K. Sycara und M. Wooldridge (Eds.), *Proceedings of the Second International Conference on Autonomous Agents*, Seiten 354–361. ACM Press, New York, 1998.

[Dorer, 1996] K. Dorer. Dynamic learning. *Diplomarbeit*, Fachhochschule Furtwangen, 1996.

[Dorer, 1999a] K. Dorer. Behavior networks for continuous domains using situation–dependent motivations. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI–99)*, Seiten 1233–1238, Stockholm, 1999. Morgan Kaufmann.

[Dorer, 1999b] K. Dorer. Extended behavior networks for the MagmaFreiburg team. In S. Coradeschi, T. Balch, G. Kraetzschmar, und P. Stone (Eds.), *Team Descriptions Simulation League*, Seiten 79–83. Linköping University Electronic Press, Stockholm, 1999.

[Dörner, 1996] D. Dörner. Eine Systemtheorie der Motivation. In J. Kuhl und H. Heckhausen (Hrsg.), *Enzyklopädie der Psychologie*, 4(4), Seiten 329 – 357. Hogrefe–Verlag, Göttingen, 1996.

[Dörner, 1999] D. Dörner. *Bauplan für eine Seele.* Rowohlt, Hamburg, 1999.

[Eisenführ und Weber, 1999] F. Eisenführ und M. Weber. *Rationales Entscheiden.* Springer Verlag, Berlin; Heidelberg, 1999.

[Erol *et al.*, 1994] K. Erol, J. Hendler, und D. S. Nau. UMCP: A sound and complete procedure for hierarchical task–network planning. In *Proceedings of the AIPS–94 Conference*, Seiten 249–254. AAAI Press, 1994.

[Fikes und Nilsson, 1971] R. E. Fikes und N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2, Seiten 189–208, 1971.

[Firby, 1989] R. J. Firby. Adaptive execution in complex dynamic worlds. *Technical Report YALEU/CSD/RR*, 672, 1989.

[Fischer *et al.*, 1995] K. Fischer, J. Müller, und M. Pischl. A pragmatic BDI architecture. In M. Wooldridge, J. Mülller, und M. Tambe (Eds.), *Intelligent Agents II*, 1037, Seiten 203–218. Springer Verlag, Heidelberg, 1995.

[Franklin *et al.*, 1996] S. Franklin, A. Graesser, B. Olde, H. Song, und A. Negatu. Virtual Mattie – an intelligent clerical agent. *AAAI Symposium on Embodied Cognition and Action*, 1996.

[Friend und Blume, 1975] I. Friend und M. Blume. The demand for risky assets. *American Economic Review*, 64, Seiten 900–922, 1975.

[Gallistel, 1990] R. Gallistel. *The Organisation of Learning.* MIT–Press, Cambridge,Mass., 1990.

[Gat, 1992] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real–world mobile robots. In *Proceedings of AAAI–92 conference*, Seiten 809–815. Morgan Kaufmann, 1992.

[Gat, 1997] E. Gat. On three–layer architectures. In D. Kortenkamp, R. P. Bonnasso, und R. Murphy (Eds.), *Artificial Intelligence and Mobile Robots.* AAAI Press, 1997.

[Ginsberg, 1989] M. L. Ginsberg. Universal planning: An (almost) universally bad idea. In *AI–Magazine*, 10(4), Seiten 40 – 44. AAAI Press, 1989.

[Goetz, 1997] Ph. S. Goetz. *Attractors in Recurrent Behavior Networks.* Ph.D. thesis, State University of New York, Buffalo, 1997.

[Gollwitzer und Malzacher, 1996] P. Gollwitzer und J. Malzacher. Absichten und Vorsätze. In J. Kuhl und H. Heckhausen (Hrsg.), *Enzyklopädie der Psychologie*, IV(4), Seiten 427–468. Hogrefe–Verlag, Göttingen, 1996.

[Goodwin und Simmons, 1992] R. Goodwin und R. Simmons. Rational handling of multiple goals for mobile robots. *Proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS–92)*, 1992.

[Haddawy und Hanks, 1998] P. Haddawy und S. Hanks. Utility models for goal–directed decision–theoretic planners. *Computational Intelligence*, 14(3), Seiten 392–433, 1998.

[Hanks, 1990] S. Hanks. *Projecting Plans for Uncertain Worlds.* Ph.D. Thesis, Yale University, NewHaven, CT., 1990.

[Hayes–Roth, 1995] B. Hayes–Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence*, 72, Seiten 329–365, 1995.

[Hayes–Roth und Hayes–Roth, 1979] B. Hayes–Roth und F. Hayes–Roth. A cognitive model of planning. *Cognitive Science*, 3, Seiten 275–310, 1979.

[Heckhausen, 1989] H. Heckhausen. *Motivation und Handeln.* 2.Auflage, Springer, Berlin, 1989.

[Hoey *et al.*, 1999] J. Hoey, R. St–Aubin, A. Hu, und C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In K. Laskey und H. Prade (Eds.), *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann Publishers, San Francisco, 1999.

[Horvitz *et al.*, 1988] E. J. Horvitz, J. S. Breese, und M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, 2, Seiten 247–302, 1988.

[Howard, 1960] R. A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, Ma., 1960.

[Howard und Matheson, 1984] R. A. Howard und J. E. Matheson. Influence diagrams. In R. A. Howard und J. E. Matheson (Eds.), *Readings on the Principles and Applications of Decision Analysis*, Seiten 721–762. Strategic Decisions Group, Menlo Park, California, 1984.

[Huffman, 1971] D. A. Huffman. Impossible objects as nonsense sentences. In B. Meltzer und D. Mitchie (Eds.), *Machine Intelligence*, 6, Seiten 295–324. Edinburgh University Press, Edinburgh, 1971.

[Jäger und Christaller, 1997] H. Jäger und Th. Christaller. Dual dynamics: Designing behavior systems for autonomous robots. In S. Fujimura und M. Sugisaka (Eds.), *Proceedings of the International Symposium on Artificial Life and Robotics*, Seiten 76–79, Beppu, Japan, 1997.

[Jensen und Veloso, 1998] R. M. Jensen und M. Veloso. Interleaving deliberative and reactive planning in dynamic multi–agent domains. In *Proceedings of the AAAI–Fall Symposium on Integraged Planning for Autonomous Agent Architectures*. AAAI Press, 1998.

[Johnson, 1997] T. R. Johnson. Control in Act-R and Soar. In M. G. Shafto und P. Langley (Eds.), *Proceedings of the Nineteenth Anual Conference of the Cognitive Science Society*, Seiten 343–348. Lawrence Erlbaum Associates, London, 1997.

[Jonsson und Bäckström, 1996] P. Jonsson und Ch. Bäckström. On the size of reactive plans. In *Proceedings of the AAAI–96 Conference*. AAAI Press, 1996.

[Jungermann *et al.*, 1998] H. Jungermann, H. R. Pfister, und K. Fischer. *Die Psychologie der Entscheidung*. Spektrum Akademischer Verlag, Heidelberg, 1998.

[Kaelbling, 1986] L. P. Kaelbling. An architecture for intelligent reactive systems. In M. P. Georgeff und A. L. Lansky (Eds.), *Reasoning About Actions and Plans*, Seiten 395–410. Morgan Kaufmann, 1986.

[Kaelbling und Rosenschein, 1990] L. P. Kaelbling und S. J. Rosenschein. Action and planning in embedded agents. In P. Maes (Ed.), *Designing Autonomous Agents*, Seiten 35–48. MIT Press, Cambridge, Mass., 1990.

[Kahnemann und Tversky, 1979] D. Kahnemann und A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47, Seiten 263–291, 1979.

[Kuhl, 1982] J. Kuhl. Action– vs. state–orientation as a mediator between motivation and action. In W. Hacker, W. Volpert, und M. Cranach (Eds.), *Cognitive and Motivational Aspects of Action*, Seiten 67–85. North Holland, Amsterdam, 1982.

[Kushmerick *et al.*, 1995] N. Kushmerick, S. Hanks, und D. S. Weld. An algorithm for probabilistic planning. *Artificial Inteligence*, 76(1-2), Seiten 239–286, 1995.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, und P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33, Seiten 1–64, 1987.

[Lalanda und Hayes–Roth, 1994] Ph. Lalanda und B. Hayes–Roth. Deadline management in intelligent agents. *Knowledge Systems Laboratory Report*, 94–27, 1994.

[Lewin *et al.*, 1944] K. Lewin, T. Dembo, L. Festinger, und P. S. Sears. Level of aspiration. In J. M. Hunt (Ed.), *Personality and the Behavior Disorders*, 1, Seiten 333–378. Ronald Press, New York, 1944.

[Littman *et al.*, 1995] M. L. Littman, T. L. Dean, und L. P. Kaelbling. On the complexity of solving markov decision problems. *Proceedings of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, 1995.

[Maes, 1989] P. Maes. The dynamics of action selection. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI–87)*, Seiten 991–997, Detroit, 1989. Morgan Kaufmann.

[Maes, 1990a] P. Maes. How to do the right thing. *Connection Science Journal*, 1(3), 1990.

[Maes, 1990b] P. Maes. Situated agents can have goals. In *Journal for Robotics and Autonomous Systems*, 6(1), Seiten 49–70, North–Holland, 1990. Elsevier Science Publishers.

[Maes, 1991] P. Maes. Adaptive action selection. In *Proceedings Thirteenth Annual Conference of the Cognitive Science Society*, Seiten 108–113, Hillsdale, 1991. Lawrence Erlbaum.

[Maes, 1992] P. Maes. Learning behavior networks from experience. In F. Varela und P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life*, Seiten 48–57. MIT–Press, Paris, 1992.

[Mataric, 1997] M. Mataric. Reinforcement learning in the multi–robot domain. *Autonomous Robots*, 4(1), Seiten 73–83, 1997.

[McAllester und Rosenblitt, 1991] D. McAllester und D. Rosenblitt. Systematic nonlinear planning. *Proceedings of the AAAI–91 conference*, Seiten 634–639, 1991.

[McFarland und Bösser, 1996] D. McFarland und Th. Bösser. *Intelligent Behavior in Animals and Robots.* MIT–Press, Cambridge, Mass., 1996.

[Müller, 1995] J. P. Müller. Modeling reactive behaviour in vertically layered agent architectures. In M. J. Wooldridge und N. R. Jennings (Eds.), *Intelligent Agents*, Seiten 261–276. Springer, Berlin, 1995.

[Müller, 1996] J. P. Müller. The design of intelligent agents: A layered approach. In *Lecture Notes in Artificial Intelligence*, 1177, Berlin, 1996. Springer.

[Navon und Gopher, 1979] D. Navon und D. Gopher. On the economy of the human processing system. *Psychological Review*, 86(3), Seiten 214–255, 1979.

[Neumann und Morgenstern, 1944] J. von Neumann und O. Morgenstern. *Theory of Games and Economic behavior*. Princeton University Press, Princeton, N.J., 1944.

[Newell, 1990] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Mass., 1990.

[Newell und Simon, 1976] A. Newell und H. A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3), Seiten 113–126, 1976.

[Noda, 1995] I. Noda. Soccer server: a simulator of RoboCup. In *Proceedings of AI symposium*, Seiten 29–34. Japanese Society for Artificial Intelligence, 1995.

[Norman, 1981] D. A. Norman. Categorization of action slips. *Psychological Review*, 88(1), Seiten 1–15, 1981.

[Norman und Bobrow, 1975] D. A. Norman und D. G. Bobrow. On data–limited and resource–limited processes. *Cognitive Psychology*, 7, Seiten 44–64, 1975.

[Norman und Shallice, 1986] D. A. Norman und T. Shallice. Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, und D. Shapiro (Eds.), *Consciousness and Self-regulation: Advances in Research and Theory*, 4. Plenum Press, New York, 1986.

[Norman, 1997] T. Norman. *Motivation–Based Direction of Planning Attention in Agents with Goal Autonomy*. Ph.D. thesis, London, 1997.

[Norman und Long, 1995] T. Norman und D. Long. Alarms: An implementation of motivated agency. In M. Wooldridge, J. Müller, und M. Tambe (Eds.), *Intelligent Agents II*, 1037, Seiten 219–234. Springer, Montreal, 1995.

[Penberthy und Weld, 1992] J. S. Penberthy und D. S. Weld. UCPOP: A sound, complete, partial–order planner for ADL. In *Proceedings of the Principles of Knowledge Representation and Reasoning conference (KR–92)*, Seiten 103–114, Cambridge, MA, 1992. Morgan Kaufmann.

[Rao und Georgeff, 1991] A. S. Rao und M. P. Georgeff. Modeling rational agents within a BDI–architecture. In R. Fikes und E. Sandewall (Eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, Seiten 473–484. Morgan Kaufmann, Cambridge, Mass., 1991.

[Rao und Georgeff, 1995] A. S. Rao und M. P. Georgeff. BDI agents: From theory to practice. *Proceedings of the First International Conference on Multiagent Systems*, Seiten 312–319, 1995.

[Rhodes, 1996] B. Rhodes. PHISH–nets: Planning heuristically in situated hybrid networks. *Masters Thesis*, 1996.

[Rosenbloom *et al.*, 1993] P. Rosenbloom, J. E. Laird, und A. Newell (Eds.). *The Soar Papers: Readings on Integrated Intelligence.* MIT Press, Cambridge, Massachusets, 1993.

[Rosenschein und Kaelbling, 1994] S. Rosenschein und L. P. Kaelbling. A situated view of representation and control. *Artificial Intelligence*, 73, 1994.

[Russel und Norvig, 1995] S. Russel und P. Norvig. *Artificial Intelligence : A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, 1995.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5, Seiten 115–135, 1974.

[Sacerdoti, 1977] E. D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier, New York, 1977.

[Saffiotti *et al.*, 1995] A. Saffiotti, K. Konolige, und E. H. Ruspini. A multivalued logic approach to integrating planning and controll. *Artificial Intelligence*, 76, 1995.

[Sahota, 1994] M. K. Sahota. Reactive deliberation: An architecture for real–time intelligent control in dynamic environments. *Proceedings of the AAAI–94 Conference*, Seiten 1303–1308, 1994.

[Schoppers, 1987] M. Schoppers. Universal plans for reactive robots. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI–87)*, Seiten 1039–1046, 1987.

[Schoppers, 1989] M. Schoppers. In defense of reaction plans as caches. In *AI–Magazine*, 10(4), Seiten 51–60. AAAI Press, 1989.

[Schoppers und Shu, 1996] M. Schoppers und R. Shu. Generalizing indexical–functional reference. *Proceedings of the AAAI–96 Conference*, Seiten 1153–1159, 1996.

[Shaffer, 1975] L. H. Shaffer. Multiple attention in continuous verbal tasks. In P. M. A. Rabbit und S. Dornic (Eds.), *Attention and Performance V.* Academic Press, New York, 1975.

[Simmons *et al.*, 1997] R. Simmons, R. Goodwin, K. Z. Haigh, S. Koenig, und J. O'Sullivan. A layered architecture for office delivery robots. In J. Müller (Ed.), *Proceedings of the Agents 97 Conference.* ACM Press, Marina del Rey, 1997.

[Simon, 1978] H. Simon. Rationality as process and as product of thought. *Journal of the American Economic Association*, 68, Seiten 1–16, 1978.

[Sloman, 1987] A. Sloman. Motives, mechanisms and emotions. *Emotion and Cognition*, 1(3), Seiten 217–234, 1987.

[Stone, 1998] P. Stone. Layered learning in multi–agent systems. *Ph.D. thesis*, Carnegie Mellon University, 1998.

[Stone *et al.*, 1999] P. Stone, M. Veloso, und P. Riley. The CMUnited–98 champion simulator team. In M. Asada und H. Kitano (Eds.), *RoboCup–98: Robot Soccer World Cup II.* Springer, Berlin, 1999.

[Stone und Veloso, 1998] P. Stone und M. Veloso. A layered approch to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence (AAI)*, 12, 1998.

[Strube, 1996] G. Strube (Hrsg.). *Wörterbuch der Kognitionswissenschaft.* Klett–Cotta, Stuttgart, 1996.

[Strube, 1998] G. Strube. Modelling motivation and action control in cognitive systems. In U. Schmid, J. Krems, und F. Wysotzki (Eds.), *Mind Modelling: A Cognitive Science Approach to Reasoning, Learning and Discovery.* Pabst Science Publishers, Lengerich, 1998.

[Tambe *et al.*, 1995] M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, und K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1), Seiten 15–39, 1995.

[Tversky, 1969] A. Tversky. Intransitivity of preferences. *Psychological Review*, 76, Seiten 31–48, 1969.

[Tversky und Kahneman, 1981] A. Tversky und D. Kahneman. The framing of decisions and the psychology of choice. *Science*, 22, Seiten 453–458, 1981.

[Tyrrell, 1994] T. Tyrrell. An evaluation of Maes' bottom–up mechanism for behavior selection. *Adaptive Behavior*, 2(4), Seiten 307–348, 1994.

[Westermann und Heise, 1996] R. Westermann und E. Heise. Motivations und Kognitionspsychologie: Einige intertheoretische Verbindungen. In J. Kuhl und H. Heckhausen (Hrsg.), *Enzyklopädie der Psychologie*, IV(4), Seiten 275–327. Hogrefe–Verlag, Göttingen, 1996.

[Winograd, 1972] T. Winograd. Understanding natural language. In *Cognitive Psychology*, 3(1), New York, 1972. Academic Press.

[Winston, 1970] P. H. Winston. Learning structural descriptions from examples. In *Technical Report MAC–TR–76*, Cambridge, Massachusetts, 1970. Massachusetts Institute of Technology.