Roboterfußball: Multiagentensystem
CS Freiburg

Diplomarbeit an der Universität Freiburg

Fakultät für Angewandte Wissenschaften

vorgelegt von:

Klaus Müller

Betreuer: Prof. Dr. B. Nebel

2000

Zusammenfassung

Die große Herausforderung des Forschungsgebietes *Roboterfußball* besteht darin, Lösungsansätze aus ganz unterschiedlichen Bereichen der Robotik und der Künstlichen Intelligenz zu einem funktionierenden Gesamtsystem zu vereinen. *Multiagentensysteme* stellen auf diesem Gebiet ein wichtiges Forschungsparadigma dar, das allgemein in letzter Zeit zunehmend an Bedeutung gewinnt. Die Roboter des *CS Freiburg* sind autonom aber kooperativ handelnde Agenten in einer dynamischen, nur ausschnittsweise wahrnehmbaren und nicht-deterministischen Umgebung. Gegenstand dieser Arbeit ist die Strategiekomponente der Agenten. Sie umfaßt einem Algorithmus zur Aktionsauswahl sowie ein Modul, das den Spielern ein kooperatives Mannschaftsspiel ermöglicht. Es wird beschrieben, wie und inwieweit der Algorithmus der *Erweiterten Verhaltensnetzwerke* von Klaus Dorer für die Aktionsauswahl der Freiburger Roboter eingesetzt werden kann. Darüber hinaus wird eine verbesserte Mannschaftstaktik auf der Basis dynamischer Rollen und dynamischer Positionspotentialfelder entwickelt. Die seit 1997 einmal im Jahr stattfindenden *RoboCup*-Wettbewerbe bieten die Möglichkeit, Erfahrungen auszutauschen und die Qualität der erstellten Systeme in spielerischem Wettstreit zu überprüfen. Die Mannschaft des *CS Freiburg* konnte bei dem diesjährigen *RoboCup* in Melbourne, Australien, den Titel des Weltmeisters erringen.

Danksagung

Ich möchte mich an dieser Stelle herzlich bei allen Mitgliedern des diesjährigen *CS-Freiburg-Teams* bedanken für die gute Zusammenarbeit trotz großer Zeitnöte. Mein besonderer Dank gilt Thilo Weigel für seine fähige Projektleitung, seine wertvollen Tips und Ratschläge, seine Mitarbeit bei der Mannschaftstaktik und seine wichtigen Hinweise zu dieser Diplomarbeit. Ebenso ein besonderer Dank an Steffen Gutmann. Ohne ihn und seinen großen Einsatz in allen Bereichen wäre das System vermutlich nicht fertig geworden. Ein besonderes Dankeschön an dieser Stelle auch an Stefan Pfeiffer und Sören Eggerstedt für das Korrekturlesen in letzter Minute. Markus Dietl gilt mein Dank für die vielen kleinen Tips u.a. zu Qt und seine stets hilfsbereite Art, wenn irgendwo Not am Mann war. Vielen Dank an Christian Reetz und Augustinus Topor für die Einweisung in das System. Bedanken möchte ich mich weiterhin bei Kornel Marko für seine Arbeit an den Grundverhalten, Boris Szerbakowski für die Betreuung der Roboter-Hardware, Maximilian Thiel für seine Bemühungen um den neuen Torwart, Burkhard Dümler für die Arbeit am Kicker-Interface, Bruno Welsch für die technische Unterstützung und nicht zuletzt bei Livia Predoiu für den Erfahrungsaustausch.

Schließlich gilt mein Dank Professor Nebel für die Ermöglichung und Betreuung dieser Arbeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Jahre 1997 wurde die *Robot World Cup Initiative (RoboCup)* gegründet [Kitano *et al.*, 1997].
Ziel war es, ein Standardproblem zu schaffen, daß Problemstellungen aus ganz unterschiedlichen Bereichen der Forschungsgebiete Robotik, Künstliche Intelligenz, Bildverarbeitung und Multiagentensysteme vereint. Die Verwendung von Standardproblemen wie z.B. Schach hat im Bereich der Künstlichen Intelligenz eine lange und fruchtbare Tradition. Hier wurde das *Fußballspiel* als Standardproblem gewählt.
Seit 1997 wird einmal jährlich eine Weltmeisterschaft im Roboterfußball veranstaltet, bei der sich die Qualität der erstellten Systeme im spielerischen Wettstreit gegeneinander zeigt. Die Weltmeisterschaft wird jeweils begleitet von einem *Workshop*, der dazu dient, die gewonnen Erkenntnisse auszutauschen und zu diskutieren. Dieses Jahr wurde der mittlerweile vierte *Robot World Cup* in Melbourne, Australien, abgehalten. Wie im letztjährigen Wettbewerb existierten vier unterschiedliche Spielklassen, die dazu dienen, bestimmte Problemstellungen besonders hervorzuheben:

- *Small-Size Robot League (F180)*
  Auf einem Feld von der Größe einer Tischtennisplatte spielen hier bis zu fünf Roboter gegeneinander. Die maximal erlaubte Grundfläche der Roboter ist auf $180\,cm^2$ begrenzt. Die Besonderheit dieser Liga ist eine über dem Spielfeld angebrachte Kamera, deren Informationen allen Spielern gleichermaßen zugänglich ist.

- *Middle-Size Robot League (F2000)*
  Im Gegensatz zur der gerade genannten sind in dieser Klasse globale Sensoren ausdrücklich verboten. Jeder Spieler besitzt seine eigenen Sensoren und kann darf daher eine Grundfläche von bis zu ca. $2000\,cm^2$ aufweisen. Den Robotern ist es möglich, per Funk unbegrenzt Informationen auszutauschen. Gespielt wird mit jeweils bis zu 4 Robotern pro Mannschaft auf einem Spielfeld der Größe 9 Meter mal 5 Meter.

- *Simulation League*
  Hier finden die Spiele in einer rein virtuellen Umgebung statt. Gespielt wird, wie im richtigen Fußball, mit jeweils elf Spielern pro Mannschaft.

- *Sony Legged Robot League*
  In dieser Klasse wird ausschließlich eine leicht abgewandelte Version des hunde-ähnlichen *AIBO* (Spielzeug-)Roboters der Firma *Sony* benutzt. Veränderungen an der Hardware sind nicht erlaubt. Die einzelnen Mannschaften unterscheiden sich daher nur in der Programmierung der Roboter.

Insgesamt läßt sich im Verlauf der letzten Jahre ein stetig wachsendes Interesse an den Wettbewerben feststellen, und zwar sowohl seitens der Medien bzw. der Öffentlichkeit als auch seitens der Forschungsgemeinschaft. Daß eine der ursprünglichen Motivationen für die Gründung der *RoboCup*-Initiative, nämlich durch die Beschäftigung mit dem Standardproblem Erkenntnisse zu gewinnen, die sich auch auf andere Bereiche übertragen lassen, dabei nicht aus den Augen verloren wird, zeigen unter anderem die im diesjährigen Workshop vorgestellten Arbeiten z.B. zum simulierten Buschfeuer-Mannagment mit Multi-Agentensystemen.

Die Abteilung *Künstliche Intelligenz* der *Albert-Ludwigs-Universität* Freiburg nimmt mit ihrer Mannschaft *CS Freiburg* seit 1998 in an den Turnieren teil. Gespielt wird in der *Real Robot Middle League*. Freiburg konnte 1998 in Paris die Weltmeisterschaft gewinnen, belegte 1999 in Stockholm den dritten Platz und errang in diesem Jahr in Melbourne erneut den Weltmeistertitel.
Gegenstand dieser Arbeit ist die sogenannte Strategiekomponente des Systems. Sie hat die Aufgabe, das Verhalten der Spieler zu steuern, indem jeweils aus einer Anzahl von Grundverhalten das der Spielsituation angemessene ausgewählt wird. Dieses übernimmt dann seinerseits die Robotersteuerung. Weiterhin ist sie für das Mannschaftsspiel der Roboter zuständig.

1.1 Zielsetzung

In dieser Arbeit soll der Fokus auf der Roboterfußballmannschaft als *Multiagentensystem* liegen. Von den Spielern des *CS Freiburg* wird gefordert, daß sie rationales, zielgerichtetes und doch reaktives sowie kooperatives Verhalten zeigen in einer dynamischen, kontinuierlichen, nicht-deterministischen Umgebung. Der von Klaus Dorer (weiter-)entwickelte Algorithmus der Erweiterten Verhaltensnetzwerke konnte in der Simulationsliga seine Tauglichkeit für eine solche Umgebung unter Beweis stellen. Ziel dieser Arbeit ist es zu zeigen, wie und inwieweit eine Strategiekomponente basierend auf Erweiterten Verhaltensnetzwerken implementiert werden kann. Darüber hinaus soll das kooperative Mannschaftsspiel verbessert werden.

1.2 Aufbau der Arbeit

Kapitel 2 gibt eine kurze Einführung in das Gebiet der Multiagentensysteme. Wesentliche Eigenschaften von Agenten und Multiagentensystemen werden genannt. Außerdem wird ein Überblick über Ansätze zur Aktionsauswahl und zur Steuerung der Agenteninteraktion gegeben.
Der so aufgespannte Rahmen dient dazu, in Kapitel 3 die Mannschaft des *CS Freiburg* und in Kapitel 4 die Ansätze einiger anderer *RoboCup*-Mannschaften einzuordnen. Kapitel 3 gibt dabei eher eine grobe Übersicht über die Hardware- und Softwarearchitektur des Freiburger Systems. Kapitel 4 geht dagegen nur auf die für diese Arbeit relevanten Ansätze zur Aktionsauswahl und Agenteninteraktion ein.
Kapitel 5 behandelt im Detail die neu gestaltete Strategiekomponente des *CS Freiburg*. Der verwendete Algorithmus der Erweiterten Verhaltensnetzwerke wird motiviert und beschrieben. Anschließend die konkrete Implementierung vorgestellt, die im Laufe dieser Arbeit entwickelt wurde. Das zweite Modul der Strategiekomponente, welches zuständig ist für die dynamische Rollenverteilung und Positionsberechnung wird anschließend erläutert. Außerdem werden Beobachtungen und Ergebnisse des *RoboCup*-Wettbewerbs geschildert.
Kapitel 6 schließlich faßt zusammen und enthält Ausblicke auf kommende Arbeiten.

Kapitel 2

Multiagentensysteme

„Agenten sind überall." So beschreibt *Flores-Mendez* die in den letzten Jahren stark gewachsene Popularität des Agentenkonzepts in der Informatik[1999]. Es ist jedoch unerwartet schwierig, eine einheitlich akzeptierte Beschreibung dessen, was einen Agenten ausmacht, zu finden. Einfachste Programme, die (z.B. vom Betriebssystem) automatisch ausgeführt werden, sobald eine bestimmte Bedingung erfüllt ist, bis hin zu komplexen Systemen, die Schlußfolgerungen über ihr eigenes Verhalten anstellen und zielgerichtet handeln, werden als Agenten bezeichnet[Rao und Wooldrige, 1999]. Dieses Kapitel soll einen kurzen Überblick über das vielgestaltige Forschungsgebiet[1] geben.

2.1 Begriffe

Russell und *Norvig* definieren einen Agenten als „anything that can be viewed as **perceiving** its environment through **sensors** and **acting** upon that environment through **effectors**."[1995]

Diese Definition ist jedoch sehr allgemein. Für viele Forscher ist eine weitere, wesentliche und grundlegende Eigenschaft von Agenten, daß sie über ein gewisses Maß an **Autonomie** verfügen. Autonomie bedeutet dabei, daß sie auf der Basis ihrer eigenen Erfahrungen handeln, ohne die Intervention von Menschen oder anderen Systemen. Ein Agent interagiert also (weitgehend) eigenmächtig in und mit seiner Umgebung, wobei diese von ganz unterschiedlicher Natur sein kann. *Russell* und *Norvig* schlagen folgende Attribute zur Klassifizierung von Umgebungen vor[1995]:

- **Diskret** vs. **kontinuierlich**: Diskrete Umgebungen lassen sich durch eine endliche Anzahl von Zuständen beschreiben. Ein Beispiel hierfür ist die

[1] Der Bereich der Forschung, der sich mit der Analyse, Konstruktion und Anwendung von Multiagentensystemen beschäftigt, wird oft als *Verteilte Künstliche Intelligenz* (engl. *distributed artificial intelligence*, *DAI*) bezeichnet.

Schach-Umgebung. Die Zustände kontinuierlicher Umgebungen enthalten dagegen auch kontinuierliche Größen wie z.B. Länge.

- **Statisch** vs. **dynamisch**: In dynamischen Umgebungen ändert sich der Zustand der Umgebung auch ohne Zutun des Agenten. Statische Umgebungen lassen dem Agenten beliebig viel Zeit zum „Nachdenken“: ihr Zustand ändert sich nur, falls der Agent eine Aktion ausführt.

- **Zugänglich** vs. **unzugänglich**: Eine Umgebung ist zugänglich, falls der Agent sämtliche ihrer Aspekte wahrnehmen kann.

- **Deterministisch** vs. **nicht-deterministisch**: Wenn sich der nachfolgende Zustand einer Umgebung ausschließlich aus dem aktuellen Zustand und der Aktion des Agenten ergibt, so wird die Umgebung als deterministisch bezeichnet.

Sei $S = \{s_1, s_2, \dots\}$ die Umgebung des Agenten, charakterisiert durch eine Menge von Zuständen, wobei sich die Umgebung zu jedem Zeitpunkt in genau einem dieser Zustände befindet. Sei P die Menge aller (diskreten) Wahrnehmungen (*Perzepte*) des Agenten und A die Menge aller Aktionen, die der Agent ausführen kann. Dann läßt sich der Agent abstrakt beschreiben durch eine Funktion
$see : S \rightarrow \wp(P)$, die Zustände der Umwelt auf die entsprechenden Perzepte abbildet[2], sowie eine Funktion
$action : \wp(P) \rightarrow A$, die Mengen von Perzepten eine Aktion zuordnet.
Das Verhalten der (nicht-deterministischen) Umgebung kann beschrieben werden durch eine Funktion
$env : S \times A \rightarrow \wp(S)$, welche den momentanen Zustand der Umgebung und die gegebene Aktion des Agenten auf die Menge möglicher, aus der Ausführung der Aktion resultierender Zustände abbildet.
Abbildung 2.1 veranschaulicht die Wechselwirkung zwischen Agent und Umgebung.

Ein Beispiel für einen (sehr einfachen) Agenten ist ein Thermostat, welches in einem Raum dafür sorgt, daß die Temperatur immer etwa einen bestimmten, vorgegebenen Wert beträgt. Die Funktion *see* reduziert hier den Zustand der Umgebung auf eine von zwei Wahrnehmungen:
$see : S \rightarrow \{temperature\_low, temperature\_ok\}$. Der Agent hat genau zwei Aktionsmöglichkeiten, nämlich die Heizung anzustellen oder abzustellen. Die Funktion *action* läßt sich damit durch zwei einfache Regeln realisieren:
$action(p) = \begin{cases} heater\_on, \ falls \ p == temperature\_low \\ heater\_off, \ sonst \end{cases}$

In komplexen Umgebungen kann man das gewünschte Verhalten eines Agenten jedoch meist nicht mehr durch eine handvoll Bedingungs-Aktions-Regeln

[2]Diese Darstellung ist natürlich stark vereinfacht. Häufig sind die Sensoren des Agenten nicht perfekt, so daß der gleiche Zustand der Umgebung zu ganz unterschiedlichen Perzeptmengen führen kann.

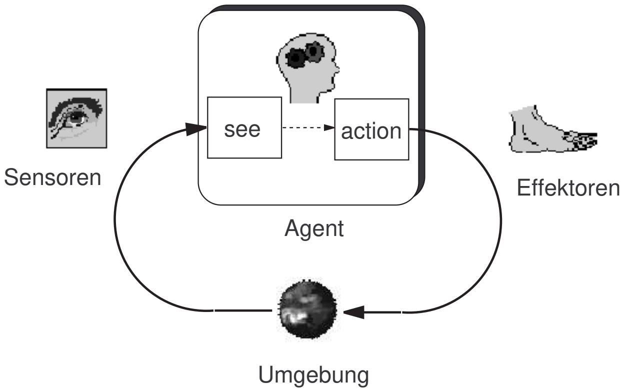

Abbildung 2.1: Interaktion des Agenten mit seiner Umgebung.

beschreiben. Trotzdem sollen Agenten auch hier **intelligentes** Verhalten zeigen. Intelligenz in diesem Sinne umfaßt drei Eigenschaften:

- **Reaktivität**: Veränderungen in der Umgebung werden von dem Agenten erkannt und rechtzeitig, in angemessener Weise darauf reagiert.
- **Pro-Aktivität**: Agenten können darüber hinaus auch selbst die Initiative ergreifen, um ihre Ziele zu erreichen bzw. ihre Aufgabe zu erfüllen.
- **soziale Fähigkeiten**: Bei der Erfüllung ihrer Aufgabe sind Agenten in der Lage mit anderen Agenten (und/oder Menschen) zu *interagieren*.

Das zentrale Problem des Agenten ist dabei das *Aktionsauswahlproblem*: *Für welche meiner möglichen Aktionen soll ich mich in diesem Moment entscheiden?*
Diese Entscheidung sollte zu jeder Zeit *rational* sein, d.h. es gilt diejenige Aktion auszuwählen, von der der Agent den meisten Nutzen erwarten kann und die ihn in die Lage versetzt, ein vorgegebenes Maß für die Qualität seines Handelns zu maximieren. Es geht vor allem auch darum, eine gute Balance zu finden zwischen reaktivem Verhalten einerseits und zielorientiertem andererseits. Wer stets nur auf die Veränderungen seiner Umgebung reagiert, riskiert, seine Ziele aus den Augen zu verlieren. Wer dagegen stur den eingeschlagenen Weg zum Ziel verfolgt, dessen Verhalten wirkt unflexibel und träge. Der nächste Abschnitt beschreibt einige Ansätze zur Lösung des Aktionsauswahlproblems.
Der Behandlung des *Interaktionsproblems*, nämlich welcher Agent wann wie mit wem interagiert, widmet sich Abschnitt 2.3.
Je nach Aufgabe und Umgebung lassen sich noch weitere wichtige Eigenschaften von Agenten nennen[Franklin und Grasser, 1996]. Hier soll vor allem noch die *Lernfähigkeit* erwähnt werden. Von den bekannten Lernparadigmen der Künstlichen Intelligenz spielt dabei das Verstärkungslernen (*reinforcement learning*) eine besonders wichtige Rolle: Der Agent nutzt hier die positivien oder negativen Rückmeldungen, die er durch die Interaktion mit seiner Umgebung bekommt, um sein Verhalten anzupassen.

Ein System aus mehreren, interagierenden, intelligenten Agenten, die gemeinsam eine bestimmte Aufgabe erfüllen, bezeichnet man als **Multiagentensystem**. Man beachte, daß die Zusammenarbeit dabei sowohl kooperativ als auch kompetitiv sein kann.
Nach *Jennings et al.* zeichnen sich Multiagentensysteme häufig dadurch aus, daß

- jeder Agent nur begrenzte Fähigkeiten besitzt und nur über unvollständige Informationen verfügt,
- die Kontrolle des Systems verteilt ist,
- Daten dezentral verwaltet werden und
- die Agenten asynchron Berechnungen durchführen[1998].

Insgesamt enthält der Begriff *Multiagentensystem* viele Freiheitsgrade. Es verwundert daher nicht, daß ein weites Spektrum ganz unterschiedlicher Anwendungen für diesen Bereich existiert:

- **Produktionsplanung**: Das System *YAMS* [Parunak, 1987] erlaubt es, den Fabrikationsprozeß einer Firma oder Organisation zu kontrollieren. Dazu werden die einzelnen Produktionseinheiten der Firma hierarchisch gruppiert und jeder der resultierenden Komponenten ein Softwareagent zugeordnet. Dieser repräsentiert die entsprechende Komponente dadurch, daß er bestimmte Fähigkeiten zur Verfügung stellt bzw. Leistungen anbietet. Die zu leistende Arbeit wird dann innerhalb des so entstehenden Multiagentensystems mit Hilfe des *Contract Net* Protokolls (siehe 2.3.2) in Form von einzelnen *tasks* verteilt.
- **Elektronische Marktplätze**: Hier werden Kaufs- und Verkaufsagenten beauftragt, bestimmte Waren zu kaufen oder verkaufen und anhand entsprechender Vorgaben zu verhandeln. *Kasbah* ist ein Beispiel für ein solches System [Chavez und Maes, 1996].
- **Informationsmanagment**: *WARREN* [Sycara *et al.*, 1998] ist ein System, das den Benutzer bei der Verwaltung seiner Geldanlagen unterstützt. Es besteht aus einer Menge von kooperativ arbeitenden Agenten, die selbstständig bestimmte Ressourcen wie Aktienkurse, Finanznachrichten usw. überwachen und wichtige Informationen zusammenstellen sowie den Benutzer bei für ihn besonders entscheidenden Entwicklungen alarmieren können.
- **Computerspiele**: Im bekannten Computerspiel *Creatures* [Grand und Cliff, 1998] kann der Benutzer den Werdegang einiger virtueller Lebewesen in ihrer simulierten Umwelt verfolgen und auch interaktiv beeinflussen. Die jeweilige Entwicklung jedes dieser Agenten wird bestimmt durch die Erfahrungen, die er im Laufe der Zeit in seiner Umwelt macht.

Abschließend sei die naheliegende Frage gestellt: Warum ist es wichtig und vorteilhaft, sich mit Multiagentensystemen zu beschäftigen?
Wichtig ist zum einen die Erkenntnis, daß intelligentes Verhalten und Interaktion sehr eng zusammenhängen. Betrachtet man den Menschen als „biologisches intelligentes System“, so beruht ein Großteil seiner intelligenten Leistungen auf hochentwickelten Fähigkeiten zu Kommunikation und Interaktion. Mit Hilfe des Paradigmas der Multiagentensysteme kann das komplexe soziale Zusammenspiel intelligenter Einheiten auf sehr natürliche Weise analysiert und modelliert werden.
Zum anderen hat man es - nicht zuletzt durch die immer größer werdende Bedeutung des Internet - immer öfter mit großen, verteilten, dynamischen, dezentral kontrollierten und heterogenen Softwaresystemen zu tun. Die Methoden der Verteilten Künstlichen Intelligenz können helfen, besser mit der diesen Systemen inhärenten Komplexität umzugehen, indem einzelne Komponenten als autonom und rational handelnde „Individuen“ analysiert und entwickelt werden.

2.2 Architekturen

Die *Architektur* eines Agenten bestimmt, nach welchen Verfahren er aus seinen Wahrnehmungen Aktionen ableitet. Die folgende Übersicht orientiert sich an [Dorer, 1999b] und [Wooldridge, 1998].

2.2.1 Logik-basierte Architekturen

Einer der ersten autonomen, intelligenten Roboter wurde Ende der sechziger Jahre am *Stanford Research Institute* (SRI) gebaut und trug den Namen „Shakey“. Seine Architektur beruhte ganz auf dem traditionellen *symbolischen* Ansatz der Künstlichen Intelligenz:
Ausgehend von einer symbolischen Repräsentation seiner Umgebung (z.B. in Prädikatenlogik 1. Stufe) sowie des gewünschten Verhaltens (für Shakey wurde das von *McCarthy* entwickelte *Situationskalkül* verwendet) *folgert* der Agent mit Hilfe eines Theorembeweisers einen *Handlungsplan*, d.h. eine Folge von Aktionen, die den Agenten ein bestimmtes Ziel erreichen läßt. Diese abstrakte Manipulation symbolischen Wissens wird auch als **Deliberation** bezeichnet.
Einerseits besticht der klassische Ansatz durch seine Klarheit und unvierselle Einsetzbarkeit. Andererseits wurde jedoch schnell klar, daß die Komplexität des Theorembeweisens zu hoch war, um in angemessener Zeit nichttriviale Handlungspläne zu generieren.
Man spricht in diesem Zusammenhang auch von *kalkulativer Rationalität*: Das obige Verfahren liefert nach Abschluß der Berechnungen eine Aktionsfolge, die optimal war zu dem Zeitpunkt, an dem die Berechnungen begannen. Es leuchtet ein, daß diese Eigenschaft in dynamischen Umgebungen wenig hilfreich sein

wird.
Es wurde im folgenden versucht, die Ausdrucksmächtigkeit der Repräsentation von Umgebung und Verhalten des Agenten einzuschränken und statt eines allgemeinen Theorembeweisers spezielle Planungsalgorithmen zu verwenden, um so effizientere Systeme zur Handlungsplanung zu erhalten.
Eines der ersten und einflußreichsten dieser Systeme war *STRIPS* [Fikes und N.Nilsson, 1971]. Im STRIPS-Formalismus werden Zustände der Umgebung beschrieben durch eine Menge von konjunktiv verknüpften Grundliteralen, d.h. evtl. negierte, lediglich auf Konstanten angewandte Prädikate. Ziele des Agenten sind Zielzustände, die auch Variablen enthalten können. Außerdem steht eine Menge von *Operatoren* zur Verfügung, spezifiziert durch

- eine *Aktionsbeschreibung*, die eine tatsächliche Aktion des Agenten repräsentiert,
- eine *Vorbedingung*, die angibt, was gelten muß, damit der Operator ausgewählt werden darf. Sie besteht aus einer Menge von konjunktiv verknüpften, positiven Literalen.
- zwei Listen von konjunktiv verknüpften, positiven Literalen, die beschreiben, was nach der Anwendung des Operators gelten soll (*add list*) bzw. nicht mehr gelten soll (*delete list*). Diese Listen werden als der *Effekt* des Operators bezeichnet.

Der Planungsalgorithmus versucht nun, eine Kette von Operatoren zu finden, die auf den Startzustand aufsetzt und alle Prädikate des Zielzustands erzeugt. Man beachte, daß dieser deliberative Ansatz eine deterministische und zugängliche Umgebung zugrundelegt. Aufgrund seiner immer noch hohen Berechnungskomplexität[3] funktioniert er außerdem nur dann gut, falls die Umgebung statisch ist oder sich nur langsam verändert sowie insgesamt eine relativ geringe Komplexität aufweist.
Neueren Planungsalgorithmen gelingt es zwar, immer komplexere oder auch nicht-deterministische Umgebungen zu meistern, grundsätzlich bleibt jedoch fraglich, ob sich in dynamischen Umgebungen der Berechnungsaufwand zur Erstellung eines Plans lohnt bzw. akzeptabel ist, weil ein einmal gefundener Plan häufig schnell wieder revidiert werden muß.

Eine weitere, prinzipielle Schwierigkeit des symbolischen Ansatzes ist, eine angemessene Repräsentation der Umgebung zu finden, die trotzdem einfach genug ist, um effiziente Deliberation zu erlauben.

[3]Nach *Bylander* gehört strips-ähnliches Planen zur Klasse der PSPACE-vollständigen Probleme[1992].

2.2.2 Reaktive Architekturen

Als Reaktion auf die grundlegenden Unzulänglichkeiten der deliberativen Ansätze wurden **reaktive** Architekturen vorgeschlagen, die auf symbolische Repräsentation und Deliberation verzichten und so eine direktere Verknüpfung von Wahrnehmung und Aktion erreichen.
Die Verfechter reakiver Architekturen prägten den Begriff des *emergenten Verhaltens*: Die einzelnen Schritte zur Erreichung eines Ziels liegen nicht mehr explizit in Form eines Plans vor, sondern ergeben sich aus der intensiven Interaktion des Agenten mit seiner Umgebung und den dabei getroffenen Aktionsauswahlentscheidungen[Maes, 1990].

Eine der ersten Architekturen dieser Art war die *Subsumptionsarchitektur* von *Brooks*[1986]. Die Aktionsauswahl wird hier gesteuert durch eine Menge von Verhalten (*behaviors*), die jeweils direkt mit den Sensoren und Aktoren verbunden sind. Sie sind darüber hinaus hierarchisch in Schichten angeordnet, wobei höhere Schichten die darunterliegenden hemmen können. Die Idee war, bei der Erstellung eines Agenten mit zunächst sehr einfachen Verhalten zu beginnen (z.B. Hindernisvermeidung), um dann darauf aufbauend immer komplexere und abstraktere Verhalten hinzuzufügen (z.B. Kartierung der Umgebung).
Brooks konnte zeigen, wie sich auf diese Weise relativ schnell Implementierungen erstellen ließen, die es Robotern in realen Umgebungen ermöglichten, einfache Verhaltensweisen (z.B. Explorieren der Umgebung) robust und effizient zu zeigen. Die Anwendung auf komplexe Aufgabenstellung ist jedoch problematisch: Je mehr Schichten bzw. Verhalten miteinander interagieren, desto schwieriger wird es, diese Interaktion nachzuvollziehen. Der Entwurf eines Agenten, dessen Verhalten also zumindest „in die richtige Richtung“ emergiert, ohne die Möglichkeit, explizit Ziele vorzugeben, ist somit äußerst kompliziert.

Weitere Reaktive Architekturen sind u.a. die *Situations-Agenten* von *Agre* und *Chapman*[1990] sowie der *Dual Dynamics* Ansatz [Jäger und Christaller, 1997].

2.2.3 BDI-Architekturen

BDI-Architekturen gehen zurück auf Arbeiten des Philosophen *Bratman* zum praktischen Schlußfolgern (*practical reasoning*) beim Menschen. Der Name BDI (*belief - desire - intention*) setzt sich zusammen aus den drei wesentlichen Elementen dieses Ansatzes: Demnach läßt sich der innere Zustand eines Agenten beschreiben durch die Informationen, die er über seine Umgebung besitzt (*beliefs*), die Handlungsmöglichkeiten, die ihm prinzipiell zur Verfügung stehen (*desires*), und diejenigen Möglichkeiten, die er sich momentan zum Ziel gesetzt hat (*intentions*).
Dabei bestimmen die aktuellen Informationen über die Umgebung und die gerade verfolgten Ziele die jeweiligen Handlungsmöglichkeiten. Der Agent überarbeitet jeweils seine Intentionen anhand der Handlungsmöglichkeiten, d.h. er setzt

sich eventuell neue Ziele, behält schon gesetzte bei oder verwirft sie. Schließlich wählt er auf der Basis seiner Intentionen eine Aktion aus.
Eine sehr erfolgreiche Implementierung einer BDI-Architektur ist das *Procedural Reasoning System* (PRS)[Georgeff und Lansky, 1987]. Das Auswählen von Aktionen erfolgt hier anhand einer vorgegebenen Bibliothek von Plänen und mit der Hilfe von Meta-Plänen steuert der Agent die Überarbeitung seiner Intentionen. Informationen über die Umgebung werden durch atomare prädikatenlogische Aussagen repräsentiert.

2.2.4 Hybride Architekturen

Da die zentrale Herausforderung des autonom handelnden Agenten darin besteht, eine gute Balance zwischen zielorientiertem und reaktivem Verhalten zu finden, versuchen moderne Architekturen, die Vorteile beider Ansätze zu vereinen.
Das Kennzeichen hybrider Architekturen ist, daß sie sowohl deliberative als auch reaktive Module enthalten. Diese Module entsprechen häufig verschiedenen Schichten. Man unterscheidet *horizontale* und *vertikale* Schichtung, je nachdem, ob jede Schicht oder nur eine der Schichten direkt die Wahrnehmungen des Agenten verarbeitet und jeweils Aktionsvorschläge generiert.

TOURINGMACHINES[Ferguson, 1995] ist ein Beispiel für eine Architektur mit horizontaler Schichtung. Eine reaktive Schicht (*reactive layer*), eine Planungsschicht (*planning layer*) und eine Modellierungsschicht (*modelling layer*, letztere zuständig für Konflikte, die sich aus dem Zusammenspiel mit anderen Agenten ergeben können) schlagen also jeweils eine Aktion vor, und ein Kontrollsystem entscheidet anhand vorgegebener Kontrollregeln, welcher der Vorschläge tatsächlich ausgeführt wird (siehe Abbildung 2.2).

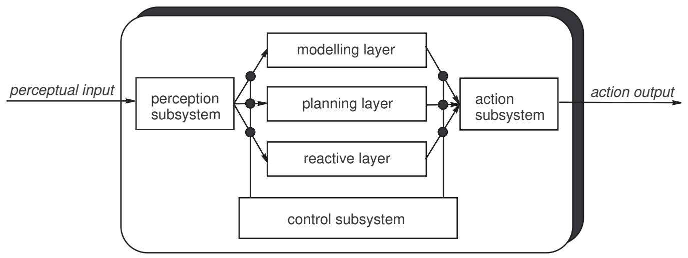

Abbildung 2.2: Hybride Architeturen: TOURINGMACHINES

Die INTERRAP-Architektur[Müller, 1997] enthält im wesentlichen die gleichen Schichten wie TOURINGMACHINES, sie sind hier jedoch vertikal verknüpft (siehe Abbildung 2.3). Zwischen den Schichten existiert zum einen ein Informationsfluß von den unteren zu den oberen Schichten, d.h. wann immer eine Schicht feststellt, daß sie nicht kompetent genug ist, die aktuelle Situation zu behandeln, aktiviert sie die darüberliegende Schicht und reicht die ihr selbst zur Verfügung stehenden Informationen weiter. Zum anderen ergibt sich ein Ausführungsfluß in umgekehrter Richtung, indem aktivierte, höherliegende Schichten der jeweils darunterliegenden entsprechende Handlungsvorgaben übergeben.

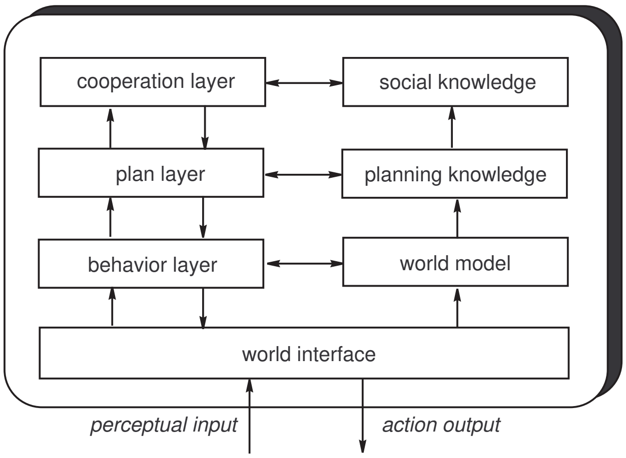

Abbildung 2.3: Hybride Architeturen: INTERRAP

Die modulare Struktur hybrider Architekturen kann die Konzeption und Implementierung von Agenten wesentlich vereinfachen. Auf der anderen Seite ergibt sich durch die Verwendung getrennter Module ein *Kohärenzproblem*, d.h. es ist keineswegs trivial, ein sinnvolles Zusammenarbeiten der einzelnen Komponenten zu gewährleisten.

2.2.5 Integrierte Architekturen

Schließlich gibt es noch einige Architekturen, die sowohl reaktive als auch zielgerichtete Aspekte in einen Mechanismus integrieren.
Die *Situated Automata* von *Rosenschein* und *Kaelbling* [Rosenschein und Kaelbling, 1990] zeichnen sich dadurch aus, daß man das Verhalten des Agenten zunächst deklarativ spezifiziert, diese Beschreibung jedoch anschließend in einen beweisbar äquivalenten digitalen Schaltkreis übersetzt wird.
Die Spezifikation erfolgt auf intuitive Weise, indem Ziele vorgegeben sowie sogenannte Zielreduktionsregeln definiert werden. Diese geben an, unter welchen Bedingungen komplexe Ziele auf evtl. mehrere, einfachere reduziert werden können. Elementare Ziele sind sog. Ausführungsziele, die direkt durch Ausführen einer bestimmten Aktion wahr gemacht werden können.
Der resultierende Schaltkreis realisiert einen endlichen Automaten, der direkt, d.h. ohne symbolische Repräsentation der Umgebung und ohne Deliberation, die Wahrnehmungen des Agenten auf Aktionen abbildet, und so eine schnelle Aktionsauswahl gewährleisten kann.

Ebenfalls zu den intergrierten Ansätzen lassen sich die *Erweiterten Verhaltensnetzwerke* von *Dorer* zählen [1999b]. Sie bilden die Grundlage für die Strategiekomponente des CS Freiburg und damit das zentrale Thema dieser Arbeit. Sie werden in Kapitel 5.3 ausführlich beschrieben.

2.3 Interaktionsmechanismen

Die Agenten eines Multiagentensystems haben zunächst ganz allgemein die Aufgabe, ihr Verhalten zu *koordinieren*, d.h. so aufeinander abzustimmen, daß zur Aufrechterhaltung des Systems möglichst wenig Eingriff von außen erforderlich wird (z.B. um Dead- oder Livelocks zu lösen). Es lassen sich zwei Arten der Koordination unterscheiden: *Kooperation*, falls die Agenten (zumindest implizit) die gleichen Ziele haben und gemeinsam ihre Erfüllung anstreben, und *Kompetition*, falls die Agenten gegensätzliche Ziele verfolgen, sich also in einem Wettstreit befinden [Huhns und Stephens, 1998].
In den nächsten Abschnitten werden einige Protokolle vorgestellt, welche die Interaktion der Agenten regeln und so koordiniertes Verhalten ermöglichen. Sie alle gehen davon aus, daß Agenten (zumindest indirekt) miteinander kommunizieren.
Natürlich ist kooperatives Verhalten prinzipiell auch ohne den Einsatz von Kommunikation möglich. Beispielsweise kann das Multiagentensystem so organisiert werden, daß jeder Agenten eine feste Rolle (funktionale Aufteilung) oder einen festgelegten Aufgabenbereich (räumliche Aufteilung) besitzt.
Eine andere Möglichkeit sind sog. *locker-room agreements*, d.h. von allen Agenten geteilte Verhaltensregeln, die in bestimmten, gemeinsam wahrgenommenen Situationen aktiviert werden[Stone und Veloso, Juni 1999]. Kommunizierende Agenten erlauben jedoch in der Regel eine flexiblere Aufgabenverteilung.

2.3.1 Blackboard Systems

Die *blackboard systems* zugrundeliegende Metapher ist die einer Gruppe von Experten, jeder von ihnen Spezialist in einem ganz bestimmten Gebiet, die gemeinsam mit Hilfe einer großen Tafel ein Problem lösen. Jeder Experte wird dann aktiv, wenn er meint, zu dem, was an der Tafel steht, etwas von seinem Wissen beitragen zu können. Auf diese Weise entsteht (im Idealfall) inkrementell eine Problemlösung.
Im einfachsten Fall entspricht die „Tafel“ eines Multiagentensystems lediglich einer gemeinsamen Datenstruktur, auf welche die Agenten nach bestimmten Regeln zugreifen. Ein Beispiel hierfür ist ein *distributed vehicle monitoring systems* (DVMS): Eine Gruppe von räumlich verteilten Agenten hat hier die Aufgabe, anhand von charakteristischen Geräuschen Vehikel zu identifizieren und ihre Bewegungen aufzuzeichnen. Dadurch, daß jeder Agent seine Beobachtungen und Hypothesen „an die gemeinsame Tafel schreibt“, können insgesamt verläßliche-

re und auch effizientere Resultate enstehen, weil Agenten überlappende Gebiete nicht doppelt zu beobachten brauchen.
Komplexere Blackboard-Systeme besitzen eine separate Kontrollkompenente, die den Lösungsprozeß steuert, indem sie die Reihenfolge der Beiträge der Agenten bestimmt[Huhns und Stephens, 1998].

2.3.2 Contract Net Protocol

Das *Contract Net* Protokoll[Davis und Smith, 1983] bietet die Möglichkeit, Aufgaben flexibel an eine Menge kooperativ problemlösender Agenten zu verteilen. Agenten fungieren dabei dynamisch als *Manager*, die bestimmte zu lösende (Teil-)Aufgaben „ausschreiben", oder als (potentielle) *Kontraktoren*, die anbieten, die Aufgabe zu lösen.
Ein Manager benachrichtigt jeweils alle potentiellen Kontraktoren über die zu lösende Aufgabe. Agenten, die in der Lage sind, die Aufgabe zu lösen, und sie lösen möchten, antworten mit einem Angebot (*bid*), das angibt, wie gut sie die Aufgabe lösen können. Aus allen eingehenden Angeboten wählt der Manager das beste aus und schließt einen „Lösungsvertrag" mit dem entsprechenden Agenten.
Läßt sich die Aufgabe weiter zerlegen, so kann der ausgewählte Kontraktor anschließend selbst zum Manager werden und Teilaufgaben ausschreiben. Auf diese Weise ergibt sich eine hierarchische Aufgabenverteilung.

Für die Anwendung in eher kompetitiven Umgebungen (z.B. Electronic Commerce) wurde das Contract Net Protokoll auf verschiedene Weise erweitert, z.B. um die Möglichkeit, Verträge unter Inkaufnahme einer entsprechenden Bestrafung zu brechen.

2.3.3 Bazaar Negotiation Model

Verhandlungen (*negotiations*) zwischen Agenten in kompetitiven Umgebungen erstrecken sich meistens über mehrere Runden. Jeweils werden Angebote und Gegenangebote ausgetauscht solange, bis eine Einigung erzielt wurde oder die Verhandlung abgebrochen wird. Nach jeder Runde muß der Agent eine Entscheidung fällen, ob er ein neues Angebot machen, akzeptieren oder sich zurückziehen möchte. In realen oder zumindest realistischen Umgebungen hat der Agent meist nur unvollständiges Weltwissen. Insbesondere die genauen Werte der internen Entscheidungsparameter anderer Agenten sind ihm unbekannt. Idealerweise kann er jeweils aus dem bisherigen Verlauf der Verhandlung und den gerade bekommenen Rückmeldungen *lernen*, also sein Wissen über bestimmte Eigenschaften seiner Verhandlungspartner anzupassen. Wie man sich leicht überlegen kann, ist es um so vorteilhafter, je mehr man über die Verhandlungspartner weiß (z.B. unter welchen Preis bei einer Verkaufsverhandlung sie auf keinen Fall gehen möchten). Motiviert durch diese Charakteristika präsentieren *Zeng* und *Sycara* einen Ansatz, der derartige Verhandlungen als sequentielle Entscheidungsprobleme modelliert[1998]. Im *Bazaar* Verhandlungsmodell wird das (dynamische) Wissen des Agenten über seine Umgebung und über seine Verhandlungspartner als Wahrscheinlichkeitsverteilung über bestimmte Parameter definiert. Die Repräsentation erfolgt in Form von *Bayes'schen Netzen.* Neue Informationen (insbesondere neue Gegenangebote) werden jeweils dazu benutzt, die Wahrscheinlichkeitsverteilung zu aktualisieren (*bayesian updating*). Adaptives Verhalten ist damit integraler Bestandteil dieses Modells. Empirische Ergebnisse legen nahe, daß die Performanz des Gesamtsystems von lernenden Agenten profitiert, daß jedoch, falls nur ein Agent lernt, dies zu Lasten der anderen Agenten und auch des Gesamtsystems geht[Zeng und Sycara, 1987].

Kapitel 3

Multiagentensystem CS Freiburg

Nachdem im vorigen Kapitel ein kurzer Überblick über das Gebiet der *Multiagentensysteme* gegeben wurde, soll im folgenden ein konkretes Multiagentensystem beschrieben werden: die Roboter-Fußballmannschaft *CS Freiburg*.

3.1 Umgebung

Wie schon zu Beginn erwähnt, spielt die Mannschaft *CS Freiburg* in der *Middle Size League*:
Zwei Mannschaften, bestehend aus bis zu drei Feldspielern und einem Torwart, treten hier auf einem Feld der Größe 9m x 5m gegeneinander an. Die Spieler dürfen dabei eine Grundfläche von bis zu 2000 cm^2 haben; der (rote) Ball hat einen Durchmesser von 20 cm. Die Benutzung globaler Sensoren ist ausdrücklich verboten.
Eine wichtige, weil oft spielentscheidende Regel, die im „menschlichen"Fußball nicht existiert, ist die sog. „10-Sekunden-Regel", die besagt, daß es den Spielern nur maximal 5 Sekunden erlaubt ist, sich im eigenen Strafraum aufzuhalten, und nur 10 Sekunden im gegnerischen. Andernfalls werden sie vom Platz gestellt, und zwar solange, bis ein Tor fällt, oder ein *Restart* veranlaßt wird[1]. Die Spielzeit beträgt zweimal 10 Minuten.
Anhand der im letzten Abschnitt genannten Attribute klassifiziert, ist die Umgebung des CS Freiburg *physikalisch*, *kontinuierlich*, *dynamisch*, *nicht-deterministisch* und *unzugänglich*.

[1] Für Details zu den Regeln siehe http://smart.informatik.uni-ulm.de/ROBOCUP/rules00/rules2000.html.

3.2 Hardware: Sensoren, Effektoren

Abbildung 3.1: Gruppenbild mit Ball.

Die Roboterplattform für die drei Feldspieler bilden die von Kurt Konolige entwickelten *Pioneer 1* Roboter.
Sie bewegen sich über zwei motorgetriebene Vorderräder und eine hinten angebrachte Laufrolle. „Von Haus aus“ bringen sie 7 Ultraschallsensoren mit. Diese reichen jedoch nicht aus, um in der Fußballumgebung zu bestehen. Deshalb besitzt jeder Spieler zusätzlich einen *Laserscanner* der Firma *SICK*, der Entfernungen zu Objekten in einem Winkelbereich von 180 Grad mit einer Genauigkeit von 1 cm und 0,5 Grad erfassen kann. Die Laserscanner werden zur Selbstlokalisation sowie zur Positionsschätzung der anderen (d.h. Mit- und Gegen-)Spieler benutzt[Gutmann *et al.*, 1999][Weigel, 1998].
Zur Ballerkennung dient eine CCD-Kamera der Firma *Newton Labs*, ausgerüstet mit dem *Cognachrome Vision System*. Die Kamera hat einen Winkelbereich von etwa +- 90 Grad[Thiel, 1999].
Darüber hinaus können die Spieler miteinander kommunizieren, und zwar über drahtlose *WaveLAN* Ethernet Verbindungen der Firma *Lucent Technologies*. Die Kommunikation läuft dabei nicht direkt, sondern über einen Zentralrechner außerhalb des Spielfelds (siehe Kapitel 3.4).
Um wirkungsvoll Fußball zu spielen, müssen die Spieler die Möglichkeit haben, den Ball zu schießen und zu führen. Bisher diente dazu ein selbstgebau-

ter *Kicker* aus Teilen des *Märklin Metallbaukastens*[Herrman, 1998]. Für die diesjährige Mannschaft wurde er ersetzt durch eine von der SICK Ausbildungswerkstatt entwickelte Schußvorrichtung, die sich zum einen durch eine höhere Schußstärke auszeichnet, zum anderen besteht jetzt die Möglichkeit, die seitlichen Führungsarme bei Bedarf hoch- und runterzufahren.
Das eigentliche Spielerprogramm läuft auf einem auf dem jeweiligen Roboter montierten *Libretto 100CT* Notebook. Steuerungsbefehle werden an den Roboter über das *Saphira* Kontrollsystem abgesetzt[Konolige *et al.*, 1997].

Der Torwart besteht im wesentlichen aus den gleichen Komponenten, allerdings sind Kicker und Laserscanner an der Querseite angebracht. Dies erlaubt ihm, sich seitlich zu bewegen und gleichzeitig das Spielfeld unmittelbar vor seinem Tor zu beobachten.

Abbildung 3.1 vermittelt einen Eindruck vom Aussehen der Spieler (hier allerdings noch mit der alten Schußvorrichtung ausgerüstet).

3.3 Spielerarchitektur

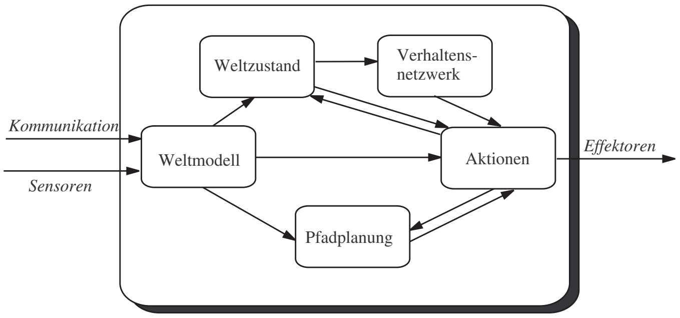

Abbildung 3.2: Architektur der Feldspieler.

Die (Software-)Architektur der Feldspieler[2] umfaßt folgende Komponenten (siehe Abbildung 3.2):

- **Weltmodell**: In einem ersten Schritt werden die Sensoreindrücke der oben beschriebenen Sensoren zu einem Modell der Fußball-Welt (aus Sicht des jeweiligen Spielers) abstrahiert. Wie in Kapitel 2.2 erwähnt, ist dieBenutzung eines expliziten Weltmodells als Voraussetzung für intelligentes

[2]Die Architektur des Torwarts ist ähnlich aufbebaut, enthält aber nur eine relativ einfache Komponente zur Verhaltenssteuerung, die in den letzten beiden Jahren weitgehend unverändert blieb. Die folgenden Ausführungen über die Module *Weltzustand*, *Verhaltensnetz* und *Aktionen* sowie das gesamte Kapitel 5 über die Strategiekomponente beziehen sich daher immer auf die Feldspieler.

Verhalten umstritten. Für den *CS Freiburg* ist es jedoch einer der Grundbausteine für das bisher sehr erfolgreiche Spiel[3][Gutmann *et al.*, 2000]. Jedes Objekt (der Spieler selbst, die Gegenspieler, seine Mitspieler und der Ball) wird durch seine Position auf dem Spielfeld repräsentiert. Dias Spielfeld wird dabei als rechteckiger Bereich in einem zweidimensionalen Koordinatensystem dargestellt, dessen Ursprung im Spielfeldmittelpunkt liegt, dessen positive x-Achse in Richtung gegnerisches Tor zeigt und dessen Einheit einem Millimeter auf dem realen Feld entspricht.
Die Positionsbeschreibung umfaßt neben x- und y-Koordinate die aktuelle Ausrichtung des Objekts[4], die aktuelle Geschwindigkeit sowie einen Zeitstempel, der angibt, wann dieser Sensoreindruck gewonnen wurde.
Weitere wichtige Informationen über das jeweilige Objekt sind seine Ausdehnung, sein Zustand (*sichtbar*, *erinnert* oder *ungültig*) und seine Identität (Ball, Gegner, Mitspieler). Es wird außerdem eine Liste fester Länge mit vorigen Objektpositionen verwaltet, u.a. zur Berechnung der aktuellen Geschwindigkeit.

- **Weltzustand**: Aufbauend auf dem Weltmodell werden die Sensordaten weiter abstrahiert und schließlich der Zustand der Fußball-Welt mit Hilfe einer Menge von reellwertigen, im Intervall [0,1] kontinuierlichen Propositionen beschrieben. Diese dienen dem *Erweiterten Verhaltensnetzwerk* als Grundlage für die Aktionsauswahl (für eine genaue Beschreibung der verwendeten Propositionen siehe Abschnitt 5.4.1).
  Einige der Propositionen lassen sich dabei mehr oder weniger direkt aus dem Weltmodell gewinnen (z.B. *Ball_in_der_Nähe*). Andere jedoch erfordern komplexere geometrische Berechnungen (z.B. *Weg_zum_Tor_frei*). Um einige dieser komplexeren Prädikate ebenfalls elegant „ablesen"zu können, wurde von *Reetz* eine erweiterte Auswertung des Weltmodells implementiert [1999].
  Sie basiert auf einer Menge von (normierten) Potentialfeldern ($f_i : \mathbb{R} \times \mathbb{R} \rightarrow [0,1]$), welche strategische Eigenschaften von Positionen des Spielfelds repräsentieren. Um die Berechnung der Potentiale nicht zu teuer werden zu lassen, wird das Spielfeld in Zellen gleicher Größe (Quadrate mit ca. 20 cm Seitenlänge) unterteilt und innerhalb dieser Zellen das Potential als konstant angenommen. Eine Abbildung $k : N \rightarrow \mathbb{R}^2$ liefert zu einem Zellindex den zugehörigen Mittelpunkt der Zelle. Folgende Potentialfelder werden benutzt:
  - f_{free}: Dieses Potentialfeld gibt an, inwieweit sich eine Position im Einflußbereich eines Gegenspielers befindet. Außerhalb des Einflußbereichs der Gegenspieler ist $f_{free}(x,y) = 1$, innerhalb eines Einflußbereichs sinkt f_{free} auf 0. Einflußbereiche von Robotern werden

[3] Aktuelles Beispiel hierfür ist die genannte *10-Sekunden-Regel*: Nur wer zu jedem Zeitpunkt ziemlich genau weiß, wo auf dem Spielfeld er sich gerade befindet und darüber hinaus wie lange schon, kann diese wichtige Regel beachten.

[4] Eine Ausrichtung von 0 Grad zeigt in Richtung positive x-Achse; positive Winkel werden gegen den Uhrzeigersinn aufgetragen.

als Ellipsen modelliert, um ihre aktuelle Geschwindigkeit und Bewegungsrichtung zu berücksichtigen. Der erste Brennpunkt der Ellipse entspricht der aktuellen Position $\vec{p}$ des Objekts, der zweite der um ein Zeitintervall Δt projezierten Position $(\vec{p'} = \vec{p} + \vec{v}\Delta t)$.

– $f_{covered}$: Eine Position erhält hier einen um so höheren Potentialwert, je mehr sie sich im Einflußbereich eines Mitspielers befindet. Der meist unterschiedlichen Wendigkeit und maximalen Beschleunigung der eigenen und der gegnerischen Roboter kann Rechnung getragen werden, indem man unterschiedliche Ellipsenradien und Projektionsintervalle Δt verwendet.

– $f_{ballview}$: Dieses Potentialfeld baut auf den beiden vorigen auf. Je mehr der direkte Weg von einer bestimmten Position zur Ballposition durch ein Objekt versperrt ist, desto niedriger ist der Wert von $f_{ballview}$. Bei der Berechnung werden jeweils Linien von der Ballposition zu allen Randpunkten betrachtet. Die Werte für die einzelnen Zellen $z_1, \ldots, z_n$ einer Linie ergeben sich rekursiv durch:
$f_{ballview}(k(z_1)) = 1$
$f_{ballview}(k(z_i)) = f_{ballview}(k(z_{i-1})) \cdot f_{free}(k(z_i)) \cdot (1 - f_{covered}(k(z_i)))$
Falls sich eine Zelle im Einflußbereich eines Mitspielers oder Gegners befindet, so sinkt der Wert für $f_{ballview}$ für diese und alle weiteren Zellen der Linie in Richtung Randpunkt. Abbildung 3.3 zeigt dieses Potentialfeld für eine Spielsituation. Rote Zellen entsprechen dabei hohen Potentialwerten, grüne niedrigen Werten. Außerdem sind gegnerische Spieler als rote Kreise dargestellt, Mitspieler als hellgrüne und der betrachtete Spieler als blauer Kreis. Man beachte den gestreckten Einflußbereich des sich zur Tormitte hin bewegenden gegnerischen Torwarts.

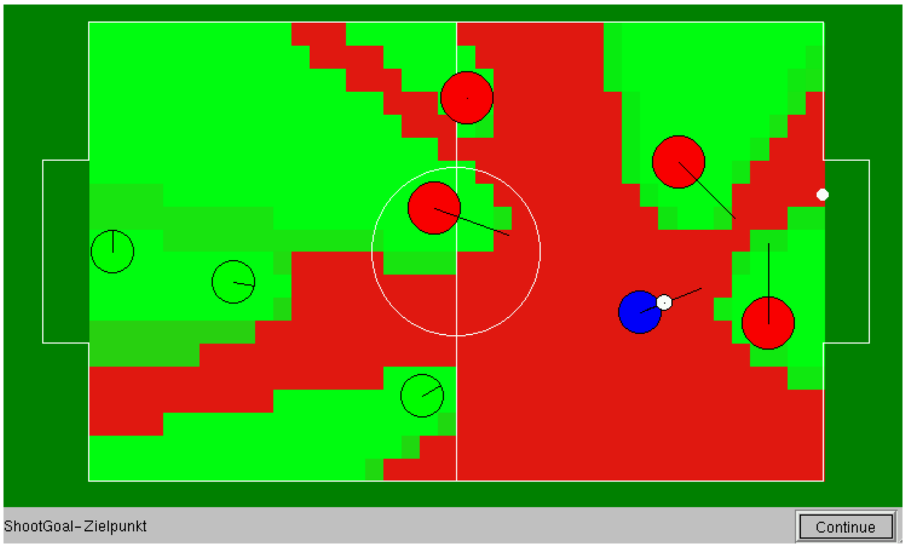

Abbildung 3.3: Das Potentialfeld $f_{ballview}$.

  - $f_{goalway}$:
    Von besonderer Bedeutung sind diejenigen Randpunkte, die sich auf der gegnerischen Torlinie befinden: Ist von ihnen aus gesehen der direkte Weg zum Ball frei, so bedeutet dies, daß für einen Roboter im Ballbesitz eine Möglichkeit zum Torschuß besteht. Das Potentialfeld $f_{goalway}$ beschränkt sich auf diese Randpunkte, wird ansonsten aber nach dem gleichen Verfahren wie $f_{ballview}$ berechnet[5].
  - $f_{desired}$: Für einen, den Angriff eines ballführenden Mitspielers unterstützenden Spieler sind bestimmte Positionen taktisch prinzipiell besser als andere. Das statische Potentialfeld $f_{desired}$ enthält diese Information. Generell ist eine Position umso besser, je näher zum gegnerischen Tor sie sich befindet. Positionen sehr weit seitlich vor dem Tor (schwieriger Torschuß), geringer Abstand zur Spielfeldbegrenzung, Positionen im gegnerischen Strafraum sowie solche vor der Tormitte (hier ist ein Behindern des ballführenden Mitspielers wahrscheinlich) werden jedoch schlechter bewertet.
  - $f_{active}, f_{support}, f_{strategic}$: Diese Potentialfelder dienen dazu, für jede Rolle, die ein Spieler einnehmen kann, die jeweils beste Position zur Erfüllung dieser Rolle zu berechnen, wiederum aufbauend auf die bisher beschriebenen Potentialfelder. Das Rollen- und Positionskonzept wird detailliert in Abschnitt 5.5 beschrieben.

  Bis auf $f_{desired}$ sind alle Potentialfelder dynamisch, werden also in jedem Zyklus neu berechnet.
  Neben der Auswertung der Propositionen hat die Komponente Weltzustand die Aufgabe, die ausgewählten Grundverhalten (s. unten) entsprechend zu parametrisieren.

- **Verhaltensnetzwerk**: Ein Erweitertes Verhaltensnetzwerk mit dem von Klaus Dorer entwickelten Algorithmus übernimmt die Verhaltensauswahl, und zwar auf der Grundlage des mit Propositionen beschriebenen Weltzustands. Auf Erweiterte Verhaltensnetzwerke wird ausführlich in Kapitel 5.3 eingegangen.
  Zusätzlich existieren für spezielle Situationen (z.B. Roboter hängt fest) *Reflexe*, d.h. Situations-Aktions-Regeln, die direkt auf das aktuelle Weltmodell angewendet werden und die Ausführung einer Aktion ohne Umweg über das Verhaltensnetzwerk zur Folge haben.

- **Aktionen**: Die eigentliche Steuerung der Roboter übernehmen eine Anzahl von Grundverhalten, die im wesentlichen über das *Saphira* Kontrollsystem die Translations- und Rotationsgeschwindigkeit des Roboters be-

[5] In der ursprünglichen Version von Reetz unterschied sich $f_{goalway}$ von $f_{ballview}$ dadurch, daß nur gegnerische Einflußbereiche betrachtet wurden unter der Annahme, daß Mitspieler den Weg für einen Torschuß freimachen würden. Diese war jedoch aufgrund der stark gestiegenen Schußkraft der Roboter, aber einer nur unwesentlich gestiegenen Beweglichkeit, nicht mehr zu halten.

einflussen (sowie dann und wann den Kicker auslösen).

Folgende Grundverhalten stehen zur Verfügung:

- **Observe-Ball**: Dieses einfache Verhalten läßt den Roboter sich jeweils Richtung Ball orientieren.
- **Search-Ball**: Hier dreht sich der Spieler um die eigene Achse, um den Ball zu suchen, und zwar zunächst in der Richtung, in welcher der Ball zuletzt gesehen wurde. Nach einer Umdrehung fährt er mit Hilfe des Pfadplaners (siehe unten) zu einer festgelegten Suchposition und dreht sich dort weiter um die eigene Achse.
- **Approach-Ball**: Dieses Verhalten dient dazu, den Spieler in Ballbesitz zu bringen, d.h. er fährt so dicht an den Ball heran, daß sich dieser direkt vor seinem Kicker und zwischen den beiden Führungsarmen befindet. Damit der Roboter beim Heranfahren den Ball mit seinen Führungsarmen nicht unabsichtlich wegstößt, werden diese zunächst hochgefahren und erst am Ende des Manövers wieder heruntergeklappt. Das Verhalten ist parametrisierbar durch Angabe der Geschwindigkeit, mit welcher der Roboter den Ball aufnehmen soll.
- **Play-Ball**: Der Spieler bewegt sich mit dem Ball auf einen bestimmten Punkt (in der Regel auf der gegnerischen Torlinie) zu, welcher vom Weltzustandsmodul übergeben wird. Die Bewegung erfolgt geradlinig, d.h. ohne Einsatz des Pfadplaners.
- **Dribble-Ball**: Wie *Play-Ball* dient dieses Verhalten dazu, den Spieler mit Ball in Richtung gegnerisches Tor zu bewegen. Die Fahrtrichtung ist jedoch nicht durch eine bestimmte Position festgelegt, sondern wird dynamisch bestimmt (siehe Abbildung 3.4): Dazu werden in jedem Zyklus eine gewisse Anzahl von Positionen in einem bestimmten Abstand und innerhalb eines bestimmten Winkelbereichs vor dem Roboter bewertet. Bewertungskriterien sind ein möglichst großer Abstand der Verbindungslinie Roboterposition → untersuchte Position zu Objekten, ein möglichst guter Winkel der neuen Roboterorientierung (falls er diese Position ansteuern würde) zum gegnerischen Tor sowie eine möglichst kleine Winkeldifferenz zwischen alter und neuer Ausrichtung. Die Bewertung einer Position ergibt sich als gewichtete Summe dieser Kriterien. Die Position mit der besten Bewertung wird als neues lokales Ziel ausgewählt und angesteuert. Dieses Verfahren ermöglicht es dem Spieler, mit Ball Objekte zu umfahren.
- **Turn-Ball**: Der Spieler ist in Ballbesitz und versucht, den Ball in einer langsamen Drehung mitzuführen und sich dabei zum gegnerischen Tor hin auszurichten.
- **Turn-Away-Ball**: Der Spieler vollzieht eine schnelle Drehung nach links oder rechts, um den Ball mit großem Winkel „wegzuschleudern“.

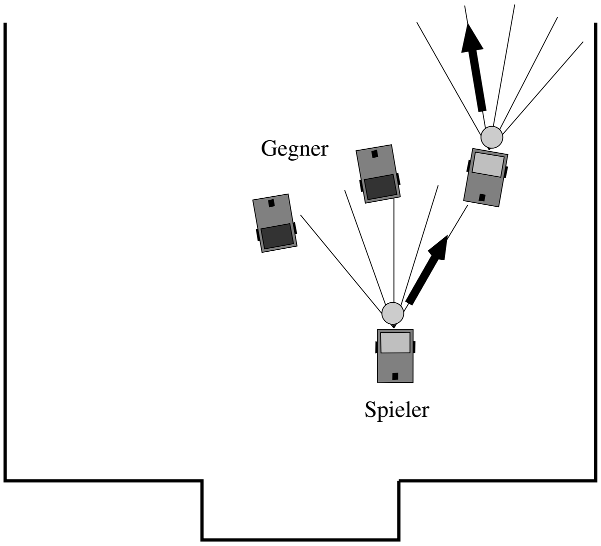

Abbildung 3.4: Lokale Zielpunkte bei Dribble-Ball.

- **Shoot-to-Pos**: Schießt den Ball zu einer bestimmten Position. Um sich möglichst exakt zur angestrebten Position hin auszurichten, fährt das Verhalten eventuell einen kleinen Bogen.
- **Quick-Shoot-to-Pos**: Wie *Shoot-to-Pos*, jedoch wird (auf Kosten der Präzision) direkt gedreht und geschossen.
- **Bump-Shoot**: Der Roboter hebt die Führungsarme, fährt mit hoher Geschwindigkeit zum Ball und löst den Kicker aus, sobald sich der Ball direkt vor ihm befindet. Liegt der Ball direkt vor dem gegnerischen Tor und ist genügend Platz vorhanden, so fährt das Verhalten einen Bogen, um den Ball ins Tor zu stoßen.
- **Flipper-Shoot**: Der Spieler stellt sich seitlich vor dem Tor des Gegners auf und orientiert sich so, daß er den Ball ins Tor katapultieren kann, wenn der Ball direkt auf ihn zurollt, indem er zum richtigen Zeitpunkt den Kicker auslöst.
- **Go-to-Position**: Fährt mit Hilfe des Pfadplaners (siehe unten) zu einer Position.
- **Free-From-Stall**: Dieses Verhalten dient dazu, den Roboter aus einer *Stall*-Situation zu befreien, d.h. wenn dieser feststellt, daß er „hängt“. Er findet dazu zunächst heraus, warum es nicht weiter geht (z.B. weil er sich zu nahe an der Bande befindet oder weil ihn ein anderes Objekt blockiert) und leitet eine entsprechende Gegenbewegung ein. Führt diese nicht zum Erfolg, so probiert er zufällige Bewegungen.

Einige der Grundverhalten wurden zu sogenannten *Compound Actions* zusammengefaßt, die aus der Sicht der Aktionsauswahl ebenfalls jeweils eine Einheit bilden.

  - **Move-Shoot-Feint**: Eine Sequenz aus den Grundverhalten *Play-Ball* und *Shoot-to-Pos* oder *Shoot-Feint*. Der Spieler bestimmt zunächst einen günstigen Punkt auf der Torlinie des Gegners und fährt mit *Play-Ball* geradewegs auf diesen zu. Bleibt der Punkt günstig, d.h. kein Gegenspieler blockiert das Tor an dieser Stelle, so wird ab einer gewissen Entfernung ein Torschuß eingeleitet (*Shoot-to-Pos*). Ansonsten hält der Spieler weiter auf den Punkt zu, sucht sich aber einen neuen Zielpunkt im gegnerischen Tor und schießt „im letzten Moment“, also aus kurzer Distanz eine Finte (*Shoot-Feint*).
  - **Dribble-KickAway-Ball**: Es wurden die Grundverhalten *Dribble-Ball*, *Shoot-to-Pos* sowie *Quick-Shoot-to-Pos* kombiniert. Zu Beginn ist *Dribble-Ball* aktiv, und zwar solange, bis keine ausreichend guten, lokalen Zielpunkte mehr gefunden werden können, z.B. weil der Spieler sich auf die Spielfeldbegrenzung zubewegt. Dann wird auf ähnliche Weise wie beim Dribbeln ein Zielpunkt für einen *Schuß* in die Nähe des gegnerischen Tores bestimmt, wobei auch Reflektionen an der Bande berücksichtigt werden. Ist noch genug Raum vorhanden, sich mittels eines kleinen Bogens präzise zum Zielpunkt auszurichten, so wird *Shoot-to-Pos* aufgerufen, ansonsten *Quick-Shoot-to-Pos*.

- **Pfadplaner**: Ein häufiges Aktionselement der Roboter ist, sich von der aktuellen Position zu einer anderen zu bewegen. Um den Weg dorthin möglichst effizient und kollisionsfrei zu gestalten, wird ein Pfadplaner benutzt [Topor, 1999]. Dieser basiert ebenfalls auf (diskretisierten) Potentialfeldern. Alle Objekte sowie die Spielfeldränder bekommen abstoßende Potentiale, der Zielpunkt jedoch anziehendes. Die Einzelpotentiale werden zu einem Gesamtpotential für das Spielfeld addiert, anschließend vom Startpunkt aus jeweils dem größten „Gefälle“(Gradienten) zum Zielpunkt gefolgt. Besondere Berücksichtigung muß dabei das Problem lokaler Minima finden.

Insgesamt wird der Bogen von Weltmodellierung bis Aktionsausführung alle 100 Millisekunden einmal durchlaufen. (Diese auch als *Zyklus* bezeichnete Zeitspanne wird vom *Saphira*-System vorgegeben.)

3.4 Teamarchitektur

Da keine globalen Sensoren erlaubt sind, kann jeder Spieler nur einen Teil des Spielfelds wahrnehmen. Allerdings sind der Kommunikation der Spieler untereinander keine Grenzen gesetzt. Die Funkverbindung wird vom Freiburger Team deshalb insbesondere dazu genutzt, aus den lokalen Weltsichten ein *globales Weltmodell* zu *fusionieren* [Dietl, 1999], und dieses den einzelnen Spielern zur Verfügung zu stellen. Diese können so also z.B. auch Objekte berücksichtigen, die sich gerade hinter ihnen befinden, wobei diese Information allerdings eine

Verzögerung von bis zu 400 ms aufweisen kann und somit weniger akkurat ist. Die Fusionierung der lokalen Weltmodelle übernimmt der Zentralrechner, dem jeder Spieler in jedem Zyklus das, was er gerade sieht, sendet, und der das globale Weltmodell dann an jeden Spieler verteilt. Man beachte, daß nur durch einen derartigen Einsatz von Kommunikation dem einzelnen Spieler die Unterscheidung zwischen „Freund und Feind“ bei den vom Laserscanner erkannten Objekten möglich wird.
Abbildung 3.5 stellt die Teamarchitektur schematisch dar. Eine weitere Aufgabe des Zentralrechners ist, eine graphische Benutzeroberfläche zur Verfügung zu stellen, mit der bestimmte Kommandos an die Spieler gesendet werden können (z.B. „Spielbeginn“ oder „Fahr zu Deiner offensiven Aufstellungsposition“). Außerdem werden alle gesendeten Daten der Spieler (lokale Weltsicht sowie u.a. gerade ausgeführte Aktion, gerade eingenommene Rolle) auf einem virtuellen Fußballfeld dargestellt und auch aufgezeichnet — ein unverzichtbares Hilfsmittel, um das Tun der Roboter zu verfolgen und zu analysieren.

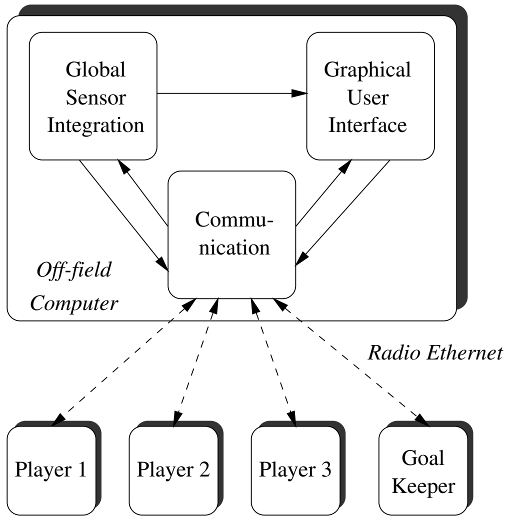

Abbildung 3.5: Die Teamarchitektur des *CS Freiburg*

Kapitel 4

Ansätze anderer Teams

Entsprechend der Zielsetzung der *RoboCup*-Initiative, ein Standardproblem für einen spielerischen Wettstreit autonomer intelligenter Systeme zu schaffen, lohnt es sich, die vielfältigen Lösungsanstze anderer Teams zu betrachten.
Die *Middle Size League*, in welcher der CS Freiburg antritt, bietet schon bei der Gestaltung der Hardware großen Spielraum. So begeisterte beispielsweise das italienische *Golem Team* beim diesjährigen Wettbewerb mit selbstentwickelten Robotern, die nicht nur omnidirektional sehen, sondern sich auch zu jeder Zeit in jede Richtung bewegen konnten.
Das Augenmerk soll hier jedoch auf den unterschiedlichen Ansätzen zur Aktionsauswahl sowie zur Agenteninteraktion liegen.

4.1 Aktionsauswahl

Berücksichtigt man die Eigenschaften der *RoboCup*-Umgebung, so überrascht nicht, daß die Aktionsauswahl der meisten Teams eher reaktiv geprägt ist. Eine weitere allgemeine Tendenz ist zu erkennen, daß in den Ligen, in denen reale Roboter zum Einsatz kommen (siehe Einleitung), eher einfachere Architekturen vorherrschen, weil hier vor allem Probleme wie der Umgang mit ungenauen Sensordaten, Selbstlokalisation, Bildverarbeitung usw. dominieren. In der Simulationsliga werden dagegen häufig komplexere Architekturen zur Aktionsauswahl erprobt (vgl. dazu Kapitel 2.2).

In der Regel werden bei RoboCup-Teams die Spielstrategien von denjenigen erstellt, die die Agentenarchitektur entworfen und implementiert haben. „Echte“ Fußballexperten werden dabei nicht mit einbezogen (siehe aber [Drücker *et al.*, 2000]).

Um die Zusammenarbeit zwischen Fußballexperten und Agentenexperten zu vereinfachen, präsentieren *Scerri* und *Ydren* für die Simulationsliga ein System, daß es Endbenutzern ermöglichen soll, mit Hilfe einer intuitiv bedienbaren gra-

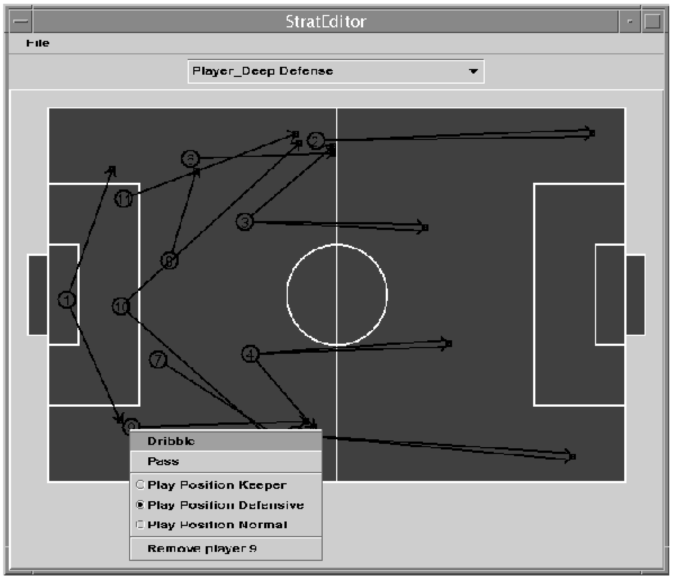

Abbildung 4.1: Erstellung der Defensivstrategie eines Teams.

fischen Oberfläche die Spielstrategie einer virtuellen Fußballmannschaft zu beeinflussen, ohne dabei um die genauen Implementierungsdetails der einzelnen Spieler wissen zu müssen[1999].
Abbildung 4.1 vermittelt einen Eindruck von den Möglichkeiten des *StratEditor*: Es stehen verschiedene Spielmodi zur Verfügung z.B. *kickoff*, *transition to attack* oder – wie in der Abbildung – *deep defense*. Für jeden Modus werden die Grundpositionen der elf Spieler festgelegt und jedem Spieler eine beliebige Anzahl von Richtungen bzw. Positionen vorgeben, in die er Dribbeln (Doppelpfeile) oder Passen (einfache Pfeile) soll. Da alle Spieler gleichzeitig bearbeitet werden, ist es einfach, ihr Verhalten aufeinander abzustimmen. Den Spielern können außerdem bestimmte „Stile“ (z.B. *crosser*: versucht immer einen Paß in die Mitte) zugeordnet werden.
Entsprechend dieser Spezifikation werden dann automatisch elf Fußballagenten erzeugt. Grundlage ist dabei eine generische Hierarchie von Verhalten, wie sie in Abbildung 4.2 zu sehen ist (jedes Rechteck entspricht dabei einem Verhalten, neben den Verhalten stehen Aktivierungsbedingungen): Die oberen Hierarchieebenen, für die Auswahl des Spielmodus (MODE) zuständig, sind dabei für alle Spieler gleich. Je nach Spielstil (ATTR) des Spielers wird dann ein bestimmter Teilbaum ausgewählt. Schließlich werden die Blätter des generischen Teilbaums, generische Aktionen der Spieler (ACTION), mit den konkreten Positions- und Richtungsvorgaben instanziiert. Diese steuern dann die Spieler zur Laufzeit.
Insgesamt konnten die Autoren zeigen, daß auch mit den Implementierungsdetails nicht vertraute Benutzer in kurzer Zeit Fußballmannschaften erstellen können. Allerdings ist die Unterscheidung zwischen Implementierungsdetail und wichigem Parameter nicht einfach. So ist beispielsweise fraglich, ob es möglich ist, gute Übergänge zwischen den einzelnen Spielmodi zu spezifizieren, ohne die Übergangsbedingung zu kennen oder beeinflussen zu können. Schade ist auch,

daß nicht schon bei der Implementierung des Systems Fußballexperten zu Rate gezogen wurden und auch eine Evaluierung des fertigen Systems mit Fußballexperten nicht erfolgte.

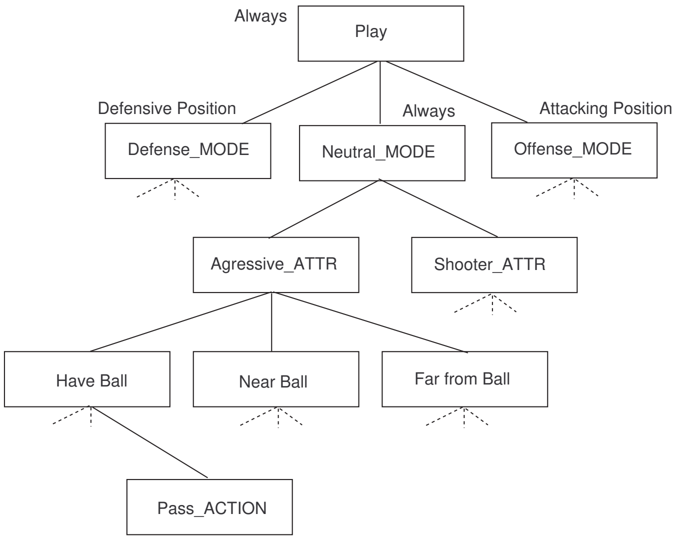

Abbildung 4.2: Generische Verhaltenshierarchie.

4.2 Agenteninteraktion

Die Fußball-Umgebung enthält sowohl kompetitive wie auch kooperative Elemente. Wettbewerb ergibt sich durch die entgegengesetzten Ziele der beiden Mannschaften, die gegeneinander antreten. Innerhalb der eigenen Mannschaft ist jedoch kooperatives Spiel gefordert. Kooperation heißt vor allem *Aufgabenteilung*. Das von den meisten (wenn nicht allen) Mannschaften hierzu verwendete Konzept ist die Einnahme und Vergabe bestimmter *Rollen*. Die Einnahme einer Rolle beeinflußt dabei mehr oder stark die weitere Aktionsauswahl des Agenten.

Ein sehr ausgefeiltes Rollenkonzept präsentieren *Stone* und *Veloso* für die Simulationsliga-Mannschaft (u.a.) der *Carnegie Mellon University*[Juni 1999]. Die einzelnen Rollen werden dabei charakterisiert durch *flexible Positionen*. Diese beinhalten jeweils eine Grundposition (*home position*) einen *Home*-Bereich und einen maximalen Bereich (siehe Abbildung 4.3). Innerhalb seines *Home*-Bereichs kann der Agent sich flexibel positionieren. Mehrere Rollen bilden zusammen eine *Formation*. Weiterhin gibt es sogenannte *Einheiten*, die sich jeweils aus mehreren Positionen zusammensetzen und die ebenfalls Teil einer Formation sind. Jede Formation besitzt einen *Kapitän*, der eine besondere Aufgabe innerhalb der Einheit wahrnimmt. Beispielsweise ordnet er jedem Spieler seiner Einheit einen Gegenspieler zu. Wichtigstes Instrument zur Koordination des Spiels ist die sogenannte „Umkleidekabinenübereinkunft“ (*locker-room agreement*). Sie wird von allen Spielern geteilt und spezifiziert u.a. alle zur Verfügung stehenden Formationen und bestimmte Regeln nach denen Positionen (d.h. Rollen) getauscht werden können. Weiterhin enthält das *locker-room agreement* vorgegebene Multiagentenpläne, die in bestimmten Situationen (z.B. bei einer Ecke) ausgeführt werden können. Koordiniertes Verhalten wird also dadurch erreicht, daß alle Spieler die gleiche Ausführungsbedingung für einen solchen Multiagentenplan besitzen. Nach dem gleichen Prinzip ist es den Spielern möglich, koordiniert die Formation zu wechseln (beispielsweise kurz vor Spielende von einer offensiven auf eine defensive, falls die eigene Mannschaft in Führung liegt). Das Konzept der *locker-room agreements* ist in der Simulationsumgebung vor allem deshalb nötig, weil hier nur sehr begrenzt Nachrichten mit nur kurzer Reichweite ausgetauscht werden können.

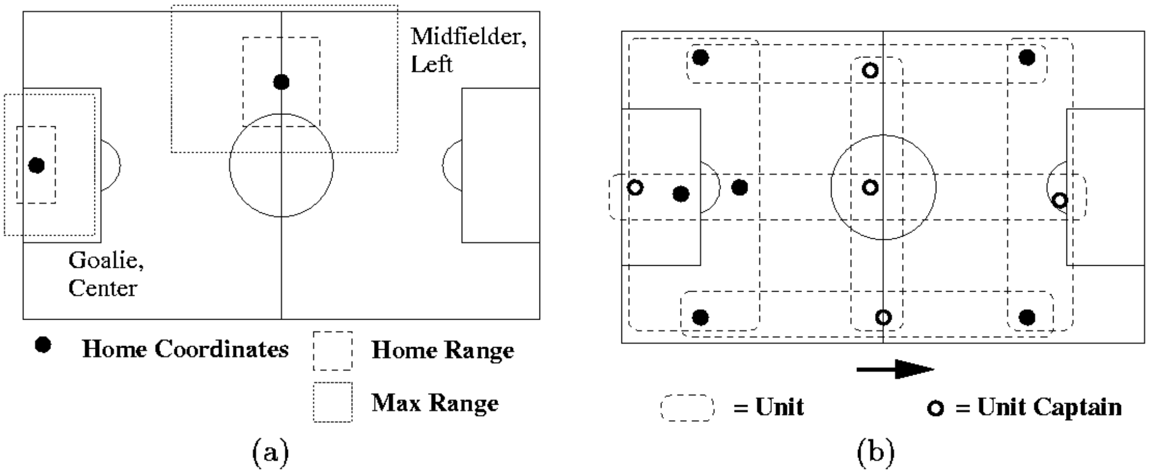

Abbildung 4.3: (a) verschiedene Positionen (Rollen) mit *home coordinate*, home range und *max range*. (b) Eine Formation.

In der Liga der Roboter mittlerer Größe ist die Kommunikation zwischen den Spielern nicht beschränkt. Auch ist normalerweise das Senden von Nachrichten über das gesamte Spielfeld möglich. *Castelpietra et al.* präsentieren für diese Liga einen kommunikationsintensiven Koordinierungsansatz, der jedoch sehr flexible Rollenwechsel ermöglicht[2000]. Er ist sehr ähnlich zu dem in der diesjährigen Strategiekomponente verwendeten Konzept und wird daher an dieser Stelle nicht weiter beschrieben (vgl. Kapitel 5.5.5). Bemerkenswert ist, daß ebenfalls unterschiedliche Formationen zum Einsatz kommen: Wenn beispielsweise der Torwart aufgrund von Lokalisierungsproblemen aus dem Tor herausgefahren ist oder er nicht mehr funktioniert, so wird der Mittelfeldspieler zu einem zweiten Verteidiger[Adorni *et al.*, 2000].

Die ebenfalls in der Simulationsliga spielende Mannschaft *Virtual Werder* führt das Konzept dynamischer Formationen noch einen Schritt weiter: Mit Hilfe eines neuronalen Netzes klassifiziert der *Online-Coach* die gerade gespielte Formation des Gegners und teilt seinen Spieler bei der nächsten Spielunterbrechung eine entsprechende Gegenformation mit, die sie einnehmen sollen[Drücker *et al.*, 2000]. Der Eingabevektor für das neuronale Netz wird dabei gewonnen, indem Zellen in einem rechteckigen Teilbereich des Spielfelds dahingehend analysiert werden, ob sich in ihnen ein Gegner befindet (Wert 1.0) oder nicht (0.0), und anschließend die Summe aller Zellen gebildet wird.

Kapitel 5

Die Strategiekomponente des CS Freiburg

Die Strategiekomponente des CS Freiburg umfaßt die Module *Weltzustand* und *Erweitertes Verhaltensnetzwerk* (siehe oben). Sie löst die zwei zentralen, in Kapitel 2 genannten Probleme des Agenten: das *Aktionsauswahlproblem* und das *Interaktionsproblem*. Es ist nützlich, sich zunächst nocheinmal notwendige und wünschenswerte Eigenschaften der Strategiekomponente in der Robocup-Welt klarzumachen (vgl. [Dorer, 1999b] und [Reetz, 1999]).

5.1 Anforderungen an die Strategiekomponente

1. Ein Fußballspiel zu gewinnen, heißt zum einen, Tore zu erzielen, zum anderen aber auch, Gegentore zu verhindern. Neben dem Gewinn des Spiels sollte gerade bei einem Wettbewerb wie RoboCup ein weiteres Ziel sein, möglichst attraktiv zu spielen. Offenbar haben Agenten in komplexen Umgebungen in der Regel mehrere, womöglich zueinander im Konflikt stehende Ziele, die je nach Situation unterschiedlich relevant sein können. Gefordert ist also eine Strategiekomponente, die mehrere Ziele gleichzeitig verfolgen und dabei auftretende Zielkonflikte auflösen kann.

2. Die hohe Dynamik der Fußball-Umgebung zusammen mit der Nicht-Determiniertheit und Unzugänglichkeit erforden eine Architektur, die es dem Spieler möglich macht, schnell auf (unvorhergesehene) Ereignisse zu reagieren. Für die Ebene der Grundverhalten ergibt sich daraus die Anforderung, diese so zu gestalten, daß das Umschalten von einem Grundverhalten zu einem anderen möglichst „reibungslos“ von statten geht.

3. Die Strategiekomponente muß berücksichtigen, daß die Sensorik kein perfektes Abbild der Umgebung liefern kann. Insbesondere bei der Ballerkennung sowie bei der Positions- und Geschwindigkeitsschätzung des Balls

ergeben sich nicht-systematische Schwankungen. Diese führen dann leicht zu einem oszillierenden Verhalten der Roboter, was wegen der Trägheit der Motoren unbedingt vermieden werden sollte. Es gilt daher, zwischen Reaktivitität auf der einen und Robustheit und *Persistenz* (d.h. die Tendenz, eine einmal begonnene Aktion auch erstmal weiterzuführen bzw. eine getroffene Entscheidung nicht sogleich wieder zu revidieren) auf der anderen Seite abzuwägen.

4. Intelligent Fußball spielen heißt, *gemeinsam* zu spielen. Wichtig ist also ein möglichst ausgeprägtes Mannschaftsspiel. Dazu gehören u.a. das Spielen von Pässen, die Einnahme bestimmter Rollen und ein gutes Stellungsspiel. Gleichzeitig sollte die Flexibiliät des Spiels jedoch möglichst wenig unter starren Rollenzuweisungen, festgelegten Grundpositionen oder Kompetenzbereichen, die der Agent nicht verlassen darf, leiden.

5. Idealerweise zeigt die Strategiekomponente ein *adaptives* Verhalten: Die Parameter der Verhaltensauswahl sowie -ausführung können anhand der im Spiel oder im Training gemachten, positiven oder negativen Erfahrungen angepaßt und so die Qualität des Verhaltens verbessert werden.

6. Die RoboCup-Welt ist eine kontinuierliche Welt. Es sollte daher möglich sein, auch Aussagen wie *gute_Möglichkeit_für_einen_Torschuß* oder *ein_Gegner_steht_vor_mir* in kontinuierlicher Form zu berücksichtigen.

7. Der Formalismus zur Beschreibung des gewünschten Verhaltens sollte möglichst intuitiv sein und eine leichte Erweiterbarkeit erlauben.

5.2 Aktionsauswahl: Wettbewerb von Aktionsmodulen

Bei der für das RoboCup-Turnier 1999 in Stockholm von Christian Reetz entwickelten Strategiekomponente wird das Aktionsrepertoire des einzelnen Spielers in Form von *Aktionsmodulen* beschrieben[1999].
Diese bestehen aus:

- einem *Aktionsnamen*,
- einer *Vorbedingung*, d.h. einer Menge von Prädikaten $\{p_1, \ldots, p_n\}$, mit $p_i \in \{true, false\}$, die jeweils das aktuelle Weltmodell auswerten. Eine Aktion ist *anwendbar*, falls die Konjunktion der Prädikate *wahr* ist.
- einer Menge von *Invarianten* $\{p'_1, \ldots, p'_m\}$, $p'_i \in \{true, false\}$. Eventuell ist es wünschenswert, eine laufende Aktion auch dann weiterzuführen, falls einige weniger wichtige Bedingungen nicht mehr gegeben sind. Invarianten werden für laufende Aktionen benutzt. Falls die Konjunktion der Invarianten *wahr* ist, so bleibt die Aktion anwendbar.

- einer *Aktivierungsfunktion* $f_{Name}(s)$, die den Weltzustand auf eine reelle Zahl abbildet und so den momentanen Nutzen der jeweiligen Aktion beschreibt.
- einer *Kontrollfunktion* $Control_{Name}$ und einer *Sequenz von Grundverhalten* $[b_1, \ldots, b_k]$. Aktionsmodule repräsentieren keine atomaren Aktionen, sondern setzen sich wiederum unter Umständen aus mehreren Grundverhalten zusammen, die in einer Sequenz angeordnet sind. Die Kontrollfunktion steuert die jeweilige Abarbeitung der Sequenz. Im einfachsten Fall werden die Grundverhalten nacheinander mit den entsprechenden Parametern aufgerufen. Möglich ist aber auch, Grundverhalten vorzeitig abzubrechen oder ganz zu überspringen, falls die Situation es erfordert.

Der *Auswahlalgorithmus* „erfragt" in jedem Zyklus von jedem Aktionsmodul dessen Aktivierungswert und wählt dasjenige mit der höchsten Aktivierung aus. Läuft die Aktion bereits, so übernimmt die jeweilige Kontrollfunktion die Steuerung. Ansonsten wird die laufende Aktion abgebrochen und die neu ausgewählte gestartet. Es besteht auch die Möglichkeit, fehlgeschlagene Aktionen zunächst solange nicht zu berücksichtigen, bis eine andere Aktion ausgeführt wurde.

Konkret verwendet Reetz in seiner Implementierung *konstante* Aktivierungsfunktionen, also eine situationsunabhängige Priorisierung der Aktionsmodule. Daß einmal aufgerufene Aktionen eine gewisse *Persistenz* zeigen, wird zum einen durch die Tatsache erreicht, daß während der Ausführung einer Aktion die weitere Ausführbarkeit durch Prüfen der meist schwächeren Invarianzbedingung anstatt der Vorbedingung festgestellt wird. Zum anderen wird der konstante Aktivierungswert während der Ausführung einer Aktion durch einen konstanten, meist höheren *Persistenzwert* ersetzt.
Als Beispiel für dieses Konzept soll das Aktionsmodul *ActionPlayBall* dienen: Es ist ausführbar, wenn der Spieler in Ballbesitz ist, sich kein anderes Objekt vor ihm befindet bzw. er zu nahe zu einer Bande steht und er eine gute Ausrichtung zum gegnerischen Tor hat. Sein konstanter Aktivierungswert beträgt 0.94 und befindet sich damit an dritter Stelle der vier „ballführenden"Aktionsmodule (*ActionPassBall*: 0.96, *ActionShootGoal*: 0.95, *ActionTurnShoot*: 0.93). Wird *ActionPlayBall* ausgewählt, d.h. es ist ausführbar, *ActionPassBall* und *ActionShootGoal* jedoch nicht, dann enspricht seine Aktivierung für die Dauer der Ausführung einem Persistenzwert von 1.1, welcher von keinem anderen Modul „überboten"werden kann. Gleichzeitig wird für die Weiterführung der Aktion lediglich die Invarianzbedingung geprüft, daß der Roboter nicht festhängt (*stalls*). Außer bei einer *Stall*-Situation hat die Ebene der Aktionsauswahl hier also keine Möglichkeit mehr, daß Verhalten des Agenten zu beeinflussen. Stattdessen entscheidet das entsprechende Grundverhalten *Move-Shoot-Feint* selbstständig, wann es zu einem Torschuß ansetzt oder den Ball wegschleudert, etwa weil ein Gegner vor dem Spieler auftaucht. Erst wenn *Move-Shoot-Feint* signalisiert, daß es erfolgreich war oder gescheitert ist, reagiert die Aktionsauswahl und wählt eine neue Aktion aus (z.B. *ActionGetBall*, um dem Ball nachzusetzen).
ActionGetBall gibt einen Eindruck von den Möglichkeiten der Kontrollfunkti-

on: Dieses Aktionsmodul beinhaltet eine Sequenz der Grundverhalten *Goto-ball* (mit Planer in die Nähe des Balls fahren) und *Approach-Ball*. Wird es ausgewählt, so startet die Kontrollfunktion zunächst *Goto-Ball* und prüft jeweils, ob schon eine gute Position erreicht wurde (Distanz und Winkel), um den Ball direkt anzufahren. Ist das der Fall, so wird *Goto-Ball* abgebrochen und stattdessen *Approach-Ball* gestartet. Da der Roboter beim Übergang vom einen zum anderen Grundverhalten nicht gestoppt wird, ergibt sich ein flüssiges Gesamtverhalten.
Das Konzept der Aktionsmodule ist insgesamt gut auf die vorhandenen, vielseitigen und mächtigen Grundverhalten zugeschnitten. Die Implementierung von Reetz trägt zu einem Spielverhalten bei, das angemessen auf die aktuelle Spielsituation reagiert, gleichzeitig den physikalischen Voraussetzungen entsprechend großen Wert auf Verhaltenspersistenz legt und im Rahmen der verfügbaren Grundverhalten einige Flexibilität zeigt.
Leider fehlt die Möglichkeit, Ziele explizit zu formulieren und in die Aktionsauswahl mit einzubeziehen. Ebenso nicht vorgesehen ist die Berücksichtigung kontinuierlicher Prädikate. Am meisten fällt meines Erachtens jedoch auf, daß die Verhaltenskontrolle recht unübersichtlich und nicht immer intuitiv über mehrere Ebenen verteilt ist. Einige Handlungssequenzen ergeben sich aus dem Wettbewerb der Aktionsmodule, andere sind vorgegeben als Teil eines Aktionsmoduls und werden von der entsprechenden Kontrollfunktion gesteuert, wieder andere sind in komplexen Grundverhalten „versteckt". Dadurch wird es schwieriger, das System zu warten und zu erweitern, und auch die Flexibilität des Gesamtverhaltens leidet darunter.
Daher sollte die diesjährige Strategiekomponente auf dem Algorithmus der Erweiterten Verhaltensnetzwerke fußen, welche im nächsten Abschnitt beschrieben werden.

5.3 Erweiterte Verhaltensnetzwerke

Die von *Klaus Dorer* vorgeschlagenen *Erweiterten Verhaltensnetzwerke* sind eine direkte Weiterentwicklung der *Verhaltensnetzwerke* von *Pattie Maes*[1989]. Die Erweiterungen von Dorer beziehen sich zum einen auf die Verwendung reellwertiger Propositionen, zum anderen auf die explizite Berücksichtigung der *Ressourcen*, die ein Verhalten benötigt, um so auch die *parallele Ausführung* von Verhalten behandeln zu können. Außerdem wurde der Mechanismus der Aktivierungsweitergabe so verändert, daß die Aktivierung eines Verhaltens stets dem *Erwartung-mal-Wert-Prinzip* der Entscheidungstheorie entspricht. Das Erwartungs-mal-Wert-Prinzip der Entscheidungstheorie besagt, daß der rational handelnde Agent stets diejenige Aktion auswählen soll, die ihm den höchsten zu erwartenden Nutzen bringt[Russell und Norvig, 1995]. Der zu erwartende Nutzen einer Aktion ist dabei definiert als die Summe über alle Produkte aus der Wahrscheinlichkeit, mit der ein bestimmter Effekt nach Ausführung der Aktion eintritt, und dem Nutzen, den dieser Effekt für den Agenten hat ($eu = \sum_i p_i \cdot u_i$).

Da die parallele Ausführung von Verhalten im Rahmen der gegenwärtigen Architektur nicht vorgesehen ist, wurde hier eine Version der Erweiterten Verhaltensnetzwerke ohne sog. Ressourceknoten implementiert (siehe auch [Dorer, 1999a]). Diese wird im folgenden beschrieben und im Abschnitt 5.3.4 auf die Eigenschaften Erweiterter Verhaltensnetzwerke eingegangen.

5.3.1 Definition

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

Ein **Erweitertes Vehaltensnetzwerk** *EBN* wird beschrieben durch ein Tupel $(\mathcal{G}, \mathcal{M}, \Pi)$, wobei gilt:

- $\mathcal{G}$ ist eine endliche Menge von *Zieltupeln* $(GCon, \iota, RCon)$, 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 dynamischen (situationsabhängigen) Wichtigkeit des Ziels. Die *Relevanz* des Ziels ergibt sich aus $r = \tau_P(RCon, s)$.
- $\mathcal{M}$ stellt eine endliche Menge von *Kompetenzmodulen* dar, wobei jedes $m \in \mathcal{M}$ charakterisiert ist durch ein Tupel $(Pre, b, Post, a)$, mit
  - $Pre \in \mathcal{L}^\wedge$ der *Vorbedingung*. Die *Ausführbarkeit* eines Kompetenzmoduls ergibt sich aus $e = \tau_P(Pre, s)$.
  - b dem Verhalten, das ausgeführt wird, wenn das Kompetenzmodul zur Ausführung selektiert wird,
  - $Post$ einer Menge von Paaren (Eff_j, ex_j), wobei $Eff_j \in \mathcal{P}$ ein nach der Ausführung des Verhaltens b erwarteter Effekt ist und $ex_j = P(Eff_j|Pre)$ die *Erwartungswahrscheinlichkeit* $\in [0..1]$ angibt, mit der die Proposition Eff gegeben Pre wahr wird.
  - a der *Aktivierung* $\in R$, welche angibt, wie wünschenswert ein Verhalten ist. a_g ist ein Vektor von Aktivierungen a_{g_i}, die das Modul direkt oder indirekt von einem Ziel g_i erhält.

- Die Menge Π enthält die *Parameter* zur Steuerung der Aktivierungsweitergabe im Netzwerk und der Aktionsauswahl, nämlich
  - $\gamma \in [0..1[$, Aktivierung durch Ziele und Nachfolger,
  - $\delta \in [0..1[$, Hemmung durch Erhaltunsziele und Konfliktoren,
  - $\beta \in [0..1[$, Trägheit der Aktivierung,
  - $\theta \in]0..\hat{a}]$, Aktivierungsschwelle mit $\hat{a}$ der maximalen Aktivierung eines Moduls,
  - $\Delta\theta \in]0..\theta]$, Abnahme der Aktivierungsschwelle.

5.3.2 Aktivierungsfluß

Die Aktivierung eines Kompetenzmoduls ist das Resultat eines Aktivierungsflusses entlang der Netzverbindungen. Aktivierungsquelle sind dabei die Ziele des Netzwerks. Ein Kompetenzmodul ist direkt mit einem Ziel verbunden, falls einer seiner Effekte Teil der Zielbedingung ist. Kompetenzmodule sind untereinander verbunden, falls ein Effekt des einen Teil der Vorbedingung des anderen ist. Die Stärke des Aktivierungsflusses wird anhand folgender Gleichungen berechnet:
Ist ein Kompetenzmodul k in der Lage, direkt zur Erfüllung eines Ziels g_i beizutragen, d.h. es existiert eine Proposition p_j, die ein Effekt des Moduls mit Erwartungswahrscheinlichkeit ex_j ist und außerdem Teil der Zielbedingung $GCon$ und die entweder beide Male negiert oder beide Male nicht negiert ist, so fließt Aktivierung zum Zeitpunkt t von

$$a_{kg_i}^{t\ \prime} = \gamma \cdot u(\iota_{g_i}, r_{g_i}^t) \cdot \nu_\gamma(p_j) \cdot ex_j. \tag{5.1}$$

$u(\iota_{g_i}, r_{g_i}^t)$ bildet dabei Wichtigkeit ι und Relevanz r des Ziels g_i auf den (normierten) Nutzen des Ziels ab[1]. Bei aktivierenden (im Gegensatz zu hemmenden) Verbindungen soll der Nutzen einer Proposition p_j um so größer sein, je mehr Propositionen der Zielbedingung bereits erfüllt sind, und je weniger die Proposition selbst erfüllt ist. Dies leistet die *Verteilungsfunktion* $\nu_\gamma(p_j) = \frac{1-\tau_P(p_j,s)}{\max(1,\sum_l(1-\tau_P(p_l,s)))}$, mit p_l den Propositionen der Zielbedingung. γ ist der globale Netzparameter, der die Stärke der Aktivierung steuert.
Steht jedoch der Effekt p_j eines Kompetenzmoduls k in direktem Gegensatz zu einer Proposition der Zielbedingung $GCon$ eines Ziels g_i, so erhält das Modul negative Aktivierung (Hemmung)

$$a_{kg_i}^{t\ \prime\prime} = -\delta \cdot u(\iota_{g_i}, r_{g_i}^t) \cdot \nu_\delta(p_j) \cdot ex_j. \tag{5.2}$$

[1] Hier wurde $u(\iota, r) = \iota \cdot r$ benutzt. (siehe [Dorer, 1999b], Seite 36)

Die Verteilungsfunktion $\nu_\delta(p_j) = \frac{\tau_P(p_j,s)}{\max(1,\sum_l(1-\tau_P(p_l,s)))}$ sorgt hier dafür, daß der Einfluß einer Proposition um so größer ist, je mehr Propositionen der Zielbedingung schon erfüllt sind (wie oben), und je *mehr* die Proposition selbst erfüllt ist.

Kann ein Kompetenzmodul k zur Erfüllung einer Vorbedingungsproposition p_{succ} eines sogenannten *Nachfolgermoduls succ* beitragen, d.h. Vorbedingungs- und Effektproposition sind entweder beide negiert oder nicht-negiert, so erhält es von diesem Aktivierung

$$a_{kg_i}^{t\ '''} = \gamma \cdot \sigma(a_{succg_i}^{t-1}) \cdot \nu_\gamma(p_{succ}) \cdot ex_{p_{succ}}. \tag{5.3}$$

Man beachte das sog. *Goaltracking*, d.h. die separate Berechnung der Aktivierung für jede Zielaktivierung $a_{succg_i}^{t-1}$ des Nachfolgermoduls zum Zeitpunkt t-1 (siehe ...). $\sigma : \mathcal{R} \rightarrow [0..1[$ ist die sog. Transferfunktion der Aktivierung; verwendet wurde $\sigma(x) = (1 + e^{-x})^{-1}$.
Ein Modul $conf$ heißt *Konfliktormodul* eines Kompetenzmoduls k, falls k einen Effekt $p_{conf} \in Pre_{conf}$ hat und genau eine der beiden Propositionen negiert ist. Die Hemmung eines Moduls durch ein Konfliktormodul beträgt

$$a_{kg_i}^{t\ ''''} = -\delta \cdot \sigma(a_{confg_i}^{t-1}) \cdot \nu_\delta(p_{conf}) \cdot ex_{p_{conf}}. \tag{5.4}$$

Erhält ein Kompetenzmodul Aktivierung von verschiedenen Propositionen desselben Ziels oder Kompetenzmoduls, so wird diese summiert. Für jedes Ziel g_i aber speichert das Kompetenzmodul k einen separaten Aktivierungswert $a_{kg_i}^t$, welcher der betragsmäßig größten, eingehenden Aktivierung entspricht:

$$a_{kg_i}^t = abs \max(a_{kg_i}^{t\ '}, a_{kg_i}^{t\ ''}, a_{kg_i}^{t\ '''}, a_{kg_i}^{t\ ''''}). \tag{5.5}$$

Die Gesamtaktivierung eines Moduls k ergibt sich schließlich durch Summation über alle Zielaktivierungen

$$a_k^t = \beta a_k^{t-1} + \sum_i a_{kg_i}^t, \tag{5.6}$$

wobei β der globale Netzparameter zur Steuerung der Trägheit der Aktivierung und damit auch der Trägheit des Verhaltens ist.

5.3.3 Aktionsauswahl

Die eigentliche Aktionsauswahl durchläuft folgenden Zyklus:

1. Berechne die Aktivierung jedes Kompetenzmoduls a_j^t nach Gleichung 5.6.
2. Verknüpfe für jedes Kompetenzmodul Aktivierung und Ausführbarkeit mit Hilfe einer monoton wachsenden Funktion $h : R \times [0,1] \to R$ zu einem *Ausführungswert*[2]. $h(a,0)$ sollte stets 0 sein, damit nur ausführbare Module ausgewählt werden.
3. Liegt das Kompetenzmodul mit dem höchsten Ausführungswert über dem Schwellwert θ, so führe das entsprechende Verhalten aus, setze θ auf seinen Ausgangswert und fahre fort bei Punkt 1.
4. Andernfalls verringere θ um $\delta\theta$ und fahre fort bei Punkt 1.

5.3.4 Eigenschaften

Das wesentliche Merkmal Erweiterter Verhaltensnetzwerke ist, daß die Aktivierung eines Kompetenzmoduls dem zu erwartenden Nutzen des zugehörigen Verhaltens in der gebenen Situation entspricht.
In Erweiterten Verhaltensnetzwerken bemißt sich der Nutzen eines Effekts einer Aktion anhand der vorgegebenen Ziele, deren Nutzen jeweils durch die *Nutzenfunktion* $u(\iota_{g_i}, r_{gi_i}^t)$ (siehe oben) beschrieben wird. Der zu erwartende Nutzen eines Kompetenzmoduls, daß ein Ziel direkt beeinflussen kann, beträgt daher $ex_j \cdot (u(\iota_{g_i}, r_{g_i}^t) \cdot \nu(p_j))$ (vgl. Gleichung 5.1 und 5.2), wobei ex_j gerade die Erwartungswahrscheinlichkeit für den Effekt p_j angibt und die Verteilungsfunktion $\nu(p_j)$ beschreibt, wie sich der Nutzen eines Ziels auf die einzelnen Zielpropositionen verteilt.
Durch die Verknüpfungen der Kompetenzmodule untereinander werden Vorbedingungen von Kompetenzmodulen zu *Unterzielen* des Netzwerks. Der Nutzen einer Proposition p_j eines solchen Unterziels ergibt sich indirekt aus dem zu erwartenden Nutzen des zugehörigen Kompetenzmoduls l und beträgt $\sigma(a_{lg_i}^{t-1}) \cdot \nu(p_j)$. Der zu erwartende Nutzen eines Moduls, welches das Unterziel beeinflussen kann, beläuft sich dann nach obigen Prinzip auf $ex_j \cdot (\sigma(a_{lg_i}^{t-1}) \cdot \nu(p_j))$ (vgl. Gleichungen 5.3 und 5.4).

Der Mechanismus der Aktivierungsweitergabe in Erweiterten Verhaltensnetzwerken unterscheidet sich in einigen wesentlichen Punkten von dem der Verhaltensnetzwerke nach Maes. Bei letzteren sind nicht nur die Ziele eines Netzwerks Aktivierungsquelle, Kompetenzmodule erhalten auch über ihre Vorbedingungen sogenannte Situationsaktivierung. Zwischen zwei Kompetenzmodulen, die miteinander verbunden sind, kann in bestimmten Situationen sowohl Situationsaktivierung in die eine, als auch Zielaktivierung in die andere Richtung fließen.

[2]Benutzt wurde $h(a,b) = a \cdot b$.

Diese Aktivierungsrückkopplung benachteiligt direkt zielerfüllende Module gegenüber solchen, die keinen direkten Effekt auf ein Ziel haben, und führen so zu einem Verstoß gegen das Erwartungs-mal-Wert Prinzip.
Des weiteren arbeiten Verhaltensnetzwerke nach Maes mit *Inputnormierung* (die eingehende Aktivierung eines Kompetenzmoduls wird durch die Anzahl Vorbedingungen bzw. erreichende/vernichtende Effekte geteilt) und einem sog. *Fan-Effekt* (die ausgehende Aktivierung von Zielen, Kompetenzmodulen und Situationspropositionen wird durch die Anzahl der empfangenden Kompetenzmodule geteilt). Beide Mechanismen führen in bestimmten Situationen ebenfalls zu einer Verletzung des Erwartungs-mal-Wert Prinzip, indem Kompetenzmodule, die mehrere Ziele gleichzeitig erreichen, nicht bevorzugt werden sowie Ziele, die durch mehrere Module erreicht werden können, benachteiligt werden.
Dorer löst diese Probleme, indem er zum einen auf Situationsaktivierung verzichtet (der Einfluß der Situation auf die Aktionsauswahl ist jetzt u.a. durch die Berücksichtigung der Ausführbarkeit eines Moduls gegeben) und zum anderen das sog. „Goaltracking“einführt: Jedes Kompetenzmodul speichert die eingehende Aktivierung nach Zielen getrennt (vgl. Gleichung 5.1 ff.). Existieren mehrere alternative Pfade von einem Modul zu einem Ziel, so wird nur der stärkste Pfad berücksichtigt, d.h. die betragsmäßig größte Aktivierung (Gleichung 5.5). Die eingehende Aktivierung von verschiedenen Zielen wird jedoch summiert (Gleichung 5.6).
Eine weitere wichtige, domänen-unabhängige Eigenschaft Erweiterter Verhaltensnetzwerke ist, daß diese bei gleichbleibender Situation stets einen stabilen Aktivierungszustand erreichen.

Um die von ihm vorgenommenen Veränderungen und Erweiterungen empirisch in spezifischen Domänezu evaluieren, erstellte Dorer außerdem eine Mannschaft für die Simulationsliga des RoboCup. Er variierte jeweils einzelne Parameter und ließ die bis auf diese Variation identischen Mannschaften jeweils in einer Serie von Spielen gegeneinander antreten[3]. Dabei kam er zu folgenden Ergebnissen:

- **situationsabhängige Ziele**: Benutzte eine der beiden Mannschaften Zielen mit statischer Wichtigkeit, d.h. ohne Relevanzbedingungen, so erzielte diese im Mittel signifikant weniger Tore als die Gegenmannschaft, deren Ziele dynamisch priorisiert werden konnten. Das Verhaltensnetzwerk beider Mannschaften enthielt dabei die drei Ziele *goal* (Wichtigkeit 1.0), *protectOwnGoal* (0.9) sowie *beUnmarked* (d.h. frei zu stehen, um anspielbar zu sein; Wichtigkeit 0.6), die bei einer der beiden Mannschaften zusätzlich die Relevanzbedingungen *inOtherHalf* bzw. *not inOtherHalf* bzw. *teammateHasBall* besaßen. Dorer führt das Ergebnis darauf zurück, daß bei der Mannschaft mit statischen Zielen die beiden wichtigeren Ziele *goal* und *protectOwnGoal* stets das weniger wichtige *beUnmarked* dominierten. Die Agenten dieser Mannschaft liefen sich in Situationen, in denen es weniger

[3]Wenn im folgenden nicht besonders erwähnt wurden die Spiele „2 gegen 2“ mit einem vereinfachten Verhaltensnetzwerk durchgeführt, um Rollenverhalten (Verteidiger, Stürmer, Abseitsfalle usw.) ausklammern zu können.

relevant war, Tore zu erzielen oder zu verhindern, deutlich seltener frei als die Spieler der Mannschaft mit dynamischen Zielen, der Ball wurde dadurch seltener abgespielt und das Spielverhalten insgesamt schlechter.

- **reellwertige vs. binäre Propositionen**: Um den Einfluß der Verwendung reellwertiger Propositionen auf die Verhaltensauswahl zu untersuchen, wurden bei einer der beiden Mannschaften die Werte sämtlicher Propositionen anhand einer festgelegten Rundungsgrenze jeweils auf 0.0 ab- oder auf 1.0 aufgerundet. Die Rundungsgrenze selbst wurde systematisch variiert. Es zeigte sich, daß bei entsprechender Optimierung der Rundungsgrenze kein signifikanter Unterschied in der mittleren Anzahl erzielter Tore beider Mannschaften feststellen ließ. Eine Analyse der reellwertigen Propositionen brachte zutage, daß bei diesen im Schnitt nur maximal 8% der Werte echt zwischen 0 und 1 lagen und ein signifikanter Effekt daher nicht zu erwarten war. In einem zweiten Experiment erweiterte Dorer das Verhaltensnetz der Mannschaften um ein Ziel, das in Konflikt stand zu den bereits vorhandenen, und um reellwertige Propositionen zur Auflösung dieses Zielkonflikts, welche in mehr als der Hälfte aller Fälle echt zwischen 0 und 1 lagen. Diesmal kam die Mannschaft mit reellwertigen Propositionen für alle untersuchten Rundungsgrenzen zu signifikant oder tendentiell signifikant besseren Ergebnissen. Noch deutlicher war der Vorteil der reellwertigen Mannschaft bei Spielen „11 gegen 11“.

- **Goaltracking**: Auch der Effekt des revidierten Mechanismus zur Aktivierungsweitergabe auf die Verhaltenskontrolle wurde untersucht. Dazu verzichtete eine der beiden Mannschaften auf das oben beschriebene Goaltracking, d.h. eingehende Aktivierung wurde immer aufsummiert, anstatt nur den stärksten Zieleinfluß zu berücksichtigen (Gleichung 5.5). In der RoboCup-Umgebung ergab sich kein signifikanter Unterschied zwischen beiden Mannschaften. Der Grund hierfür lag laut Dorer in der Struktur des Verhaltensnetzwerks begründet: Bei insgesamt nur 45 Aktivierungsverbindungen ist zu erwarten, daß Verletzungen des Erwartungs-mal-Wert-Prinzips durch Aktivierungsrückkoppelung und Aktivierungszusammenfluß nur unwesentlich zum Tragen kommen. Um diese These zu belegen, betrachtete Dorer noch eine zweite Domäne, die bekannte *Block-Welt*. Das Verhaltensnetz war hier insgesamt deutlich größer und enthielt wesentlich mehr Aktivierungsverbindungen (650 für eine Aufgabe mit 4 Blöcken). Darüber hinaus ist die Block-Welt eine statische Domäne, in der ausschließlich ziel-orientiertes Handeln eine Rolle spielt. Wie vermutet, zeigten sich diesmal deutliche Unterschiede: Dorer variierte systematisch die globalen Netzparameter δ und γ und beobachtete, ob das Verhaltensnetzwerk mit oder ohne Goaltracking die gestellte Aufgabe lösen konnte. Bei einem Problem mit vier Blöcken war dies ohne Goaltracking nur für eine von 121 Einstellung möglich. Mit Goaltracking dagegen war das Verhaltensnetzwerk bei in etwa einem Viertel der Parametervariationen erfolgreich.

- **Vergleich mit den Verhaltensnetzwerken nach Maes**: Beim direkten Vergleich der beiden Verfahren zur Aktionsauswahl waren die Erweiterten Verhaltensnetzwerke in der Domäne *RoboCup* eindeutig überlegen. Bei Spielen sowohl mit zwei gegen zwei als auch mit elf gegen elf Agenten erzielte die Mannschaft mit diesem Algorithmus im Mittel signifikant mehr Tore als ihre Gegner, die von den Verhaltensnetzwerken nach Maes gesteuert wurden.

Daß Erweiterte Verhaltensnetzwerke in kontinuierlichen, dynamischen und nicht-deterministischen Umgebungen wie dem virtuellen Fußball die Grundlage für erfolgreich und rational handelnde Agenten sein können, zeigt nicht zuletzt die Tatsache, daß das „MagmaFreiburg Soccer Team“ beim Robocup'99 in Stockholm den Titel des Vizeweltmeisters der Simulationsliga erringen konnte. In den folgenden beiden Abschnitten wird beschrieben, wie und inwieweit dieser Algorithmus für eine Mannschaft in der Liga realer Roboter mittlerer Größe Verwendung finden kann.

5.4 Ein Verhaltensnetzwerk für den CS Freiburg

Die Implementierung eines konkreten Verhaltensnetzwerks für *RoboCup 2000* erwies sich als schwieriger als erwartet. Innerhalb einer insgesamt relativ knappen Vorbereitungszeit wurden neben der Weiterentwicklung der Strategiekomponente durch den Autor u.a. die Ebene der Grundverhalten, auf welche die Aktionsauswahl aufbaut, neu strukturiert sowie neue Grundverhalten hinzugefügt. Gleichzeitig wurde die Hardware der Roboter von der Ausbildungswerkstatt der Firma SICK umgebaut, wodurch erst spät alle Roboter zur Verfügung standen. Die neuen Möglichkeiten der Erweiterten Verhaltensnetzwerke intensiv auszuprobieren und gleichzeitig innerhalb kurzer Zeit ein funktionsfähiges, robustes Gesamtsystem zu erhalten, das sich mit der beträchtlichen Spielstärke des bestehenden messen mußte, war nicht möglich. Das realisierte Netzwerk orientiert sich daher im wesentlichen an dem Aktionsauswahlkonzept des Vorjahres. Einige der aufgetretenen Probleme, die daraus gewonnenen Erkenntnisse sowie Verbesserungs- und Erweiterungsmöglichkeiten werden in Abschnitt 5.4.3 diskutiert.

Abbildung 5.1 gibt einen Eindruck von der Gesamtstruktur des Netzwerks. Es enthält insgesamt 2 Ziele, 17 Kompetenzmodule, 48 Propositionen (von denen einige aus Gründen der Übersichtlichkeit mehrfach dargestellt sind) und 21 ausschließlich aktivierenden (Effekt-)Verbindungen. Man erkennt verschiedene Gruppen von Kompetenzmodulen, u.a. die „ballführenden“ mit der gemeinsamen Vorbedingung *have_ball* und die „kooperativen“ mit dem Effekt *cooperate*. Die globalen Netzparameter (vgl. Abschnitt 5.3.1) wurden wie folgt festgelegt: Da nur aktivierende Verbindungen benutzt wurden, konnte δ entfallen (Wert 0). Verhaltenspersistenz wurde mit anderen Mitteln sichergestellt, daher erhielt der Trägheitsparameter β mit 0.1 einen nur kleinen Wert. $\gamma = 0.5$ $\theta = 0.5$ und $\Delta\theta$

= 0.1 wurden so aufeinander abgestimmt, daß die Schleife zur Aktionsauswahl immer ausreichend oft durchlaufen wird, bevor die Entscheidung für ein Kompetenzmodul fällt (vgl. Abschnitt 5.3.3). In den folgenden Abschnitten werden zunächst die verwendeten Propositionen und Ziele, anschließend die einzelnen Kompetenzmodule und ihre Verknüpfungen beschrieben.

5.4.1 Propositionen und Ziele

Die im Netzwerk vorkommenden Propositionen lassen sich grob in drei Gruppen einteilen: Zielpropositionen, allgemeine Propositionen und solche, die für ein bestimmtes Kompetenzmodul spezifisch sind. Einige der den Propositionen zugrundeliegende Wahrnehmungen sind inhärent binär (z.B. *ball_sichtbar*) und die entsprechenden Propositionen daher entweder *wahr* (Wert 1) oder *falsch* (Wert 0).

Propositionen mit echt kontinuierlichem Wertebereich werden häufig mit Hilfe der sogenannten *Fuzzy-Funktionen* des Saphira-Systems erzeugt, die ein bestimmtes kontinuierliches Maß (z.B. Distanz des Spielers zum Ball) anhand eines vorgebenen unteren und oberen Schwellwerts auf den Wertebereich [0..1] abbilden. Abbildung 5.2 zeigt die beiden Funktionen *up_straight* und *straight_down*.

Die beiden Zielpropositionen **soccergoal** und **cooperate** charakterisieren die beiden *Ziele* des Netzwerks. Sie sind „künstliche“ Propositionen in dem Sinne, daß sie immer unerfüllt bleiben, auch wenn z.B. Tore fallen. Die Wichtigkeit beider Ziele beträgt 1, sie unterscheiden sich jedoch in ihrer Relevanz: Genauer gesagt bewirkt die konträre Relevanzbedingung *role_active* im Falle *soccergoal* gegenüber **not** *role_active* bei *cooperate*, daß genau eines der beiden Ziele einen Nutzen von 1 hat und somit maximale Aktivierung „aussendet“, während von dem anderen keine Aktivierung ausgeht, weil es einen Nutzen von 0 besitzt.
Die Möglichkeit, Ziele in Erweiterten Verhaltensnetzwerken dynamisch zu priorisieren, wird im obigen Netzwerk also dazu benutzt, den Einfluß der jeweiligen Rolle des Spielers (siehe Kapitel 5.5.2) auf die Aktionsauswahl zu beschreiben. So ist zum Beispiel ein Kompetenzmodul, das nur den Effekt *cooperate* besitzt, für einen aktiven Spieler (d.h. *role_active* ist für ihn *wahr*) zwar unter Umständen ausführbar, aber gänzlich uninteressant, weil es zur Erfüllung des für ihn relevanten Ziels *soccergoal* nichts beitragen kann.

Allgemeine Propositionen beschreiben u.a. abstrakte Eigenschaften der Ballposition und die momentane Rolle des Spielers:

- **ball_present**: ist wahr, falls der Spieler selbst oder irgendein Mitspieler den Ball sieht.
- **ball_closeto_border**, **ball_near_own_goal**, **ball_in_own_half**, **ball_in_opp_penalty_area**: Die ersten beiden Propositionen sind um so mehr erfüllt, je geringer der Abstand des Balls zur nächsten Spielfeldbegrenzung bzw. zum eigenen Tor ist, die letzten beiden

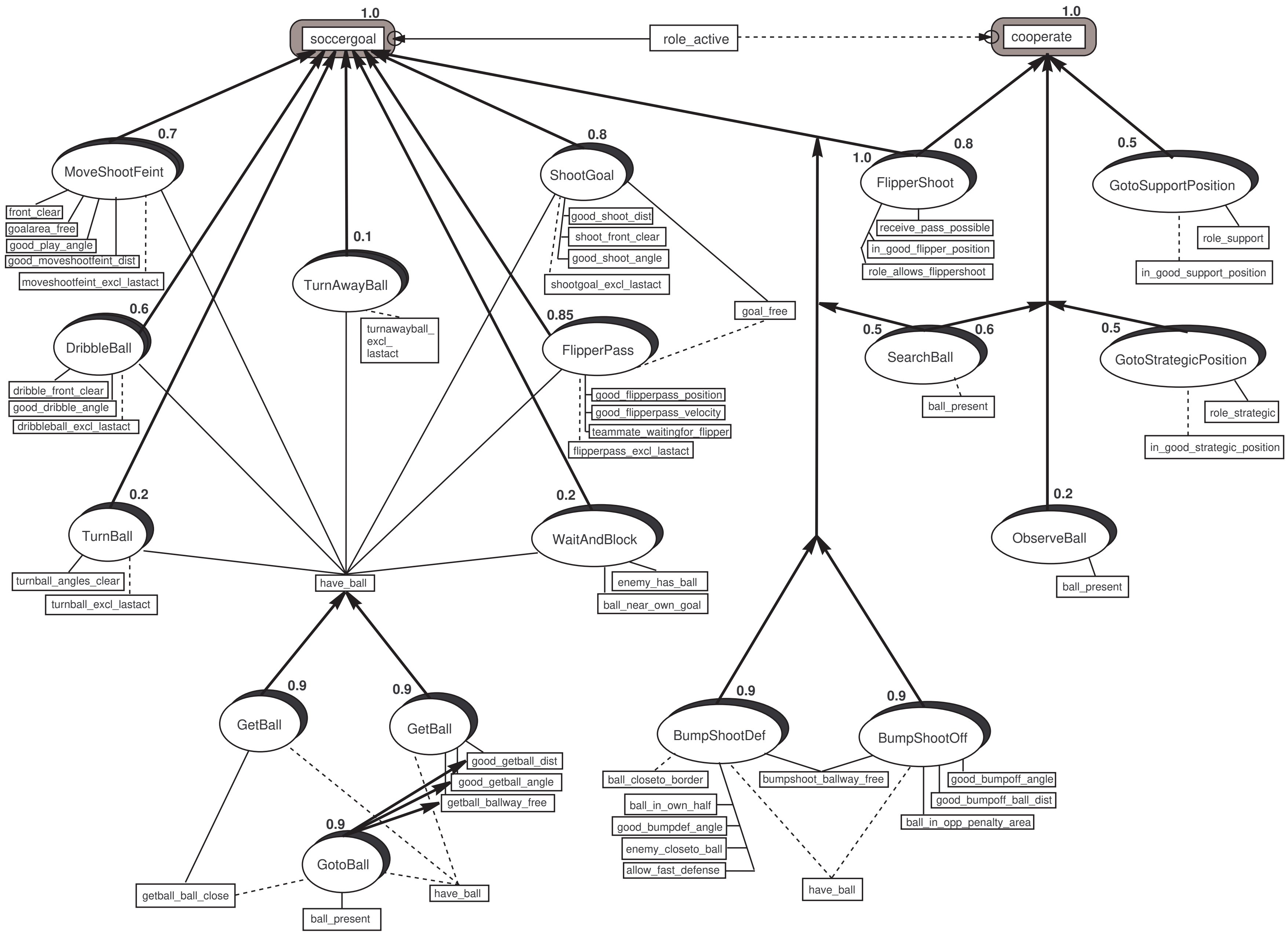

Abbildung 5.1: Das Verhaltensnetzwerk des CS Freiburg 2000

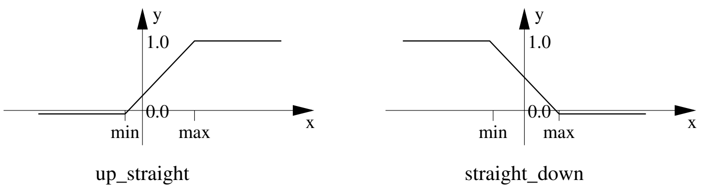

Abbildung 5.2: Die Funktionen up_straight und straight_down

genau dann, wenn sich der Ball in der eigenen Hälfte bzw. im gegnerischen Strafraum befindet.

- **have_ball**, **enemy_has_ball**: Der Spieler ist in Ballbesitz, falls der Ball direkt vor ihm liegt. Da die Orientierung der Gegenspieler in der Regel nicht bekannt ist (oder diese einen omnidirektionalen Antrieb besitzen), ist die zweite Proposition immer dann *wahr*, falls sich der Ball innerhalb eines Mindestabstands von einem der Gegenspieler befindet. Die Proposition *have_ball* nimmt eine grundlegende Klassifikation der Spielsituation aus der Sicht des aktiven Spielers vor. Sie wird außerdem zum zentralen Unterziel des Verhaltensnetzwerks, falls das Ziel *soccergoal* relevant, der aktive Spieler aber nicht in Ballbesitz ist (siehe Abbildung 5.1).

- **enemy_close_to_ball**: Diese Proposition ist wahr, falls ein Gegenspieler näher zum Ball steht als ein bestimmter Schwellwert. Dieser Schwellwert hängt dabei ab von der Distanz des Spielers selbst zum Ball.

- **role_active**, **role_support**, **role_strategic**: Je nach momentaner Rolle des Spielers ist genau eine der Propositionen wahr.

Die Gruppe der Propositionen, die spezifisch für bestimmte Kompetenzmodule sind, wird im nächsten Abschnitt zusammen mit den jeweiligen Modulen erläutert. Sie umfaßt z.B. *good_shoot_angle*, *good_play_angle* und *good_dribble_angle*: Die Anforderungen an die Orientierung des Roboters sind unterschiedlich, je nachdem welche Aktion ausgeführt werden soll. Um die Erstellung des Netzwerks zu vereinfachen, wurde für die meisten dieser Propositionen auf einen echt kontinuierlichen Wertebereich verzichtet.

5.4.2 Kompetenzmodule

Die meisten Kompetenzmodule sind direkt mit einer der beiden oder beiden Zielpropositionen verknüpft. Die Erwartungswahrscheinlichkeiten dieser Effektverbindungen wurden nicht empirisch bestimmt, sondern nach einem ähnlichen Prinzip festgelegt wie die konstanten Aktivierungswerte der in Abschnitt 5.2 beschriebenen Aktionsmodule: Dazu werden jeweils Paare von Effektverbindungen betrachtet, die von unterschiedlichen Kompetenzmodulen ausgehen. Erreichen beide Verbindungen unterschiedliche Ziele, so kann die Erwartungswahrscheinlichkeit (im folgenden: *Effektstärke*) unabhängig voneinander gewählt werden.

Ansonsten muß eine geeignete Priorisierung der Kompetenzmodule festgelegt werden, falls ihre Vorbedingungen sich nicht gegenseitig ausschließen, d.h. falls die Menge der Situationen, in denen das eine Kompetenzmodul ausführbar ist, einen nichtleeren Schnitt mit der entsprechenden Menge des anderen aufweist.

Suchen und Beobachten des Balls

Das Kompetenzmodul **ObserveBall**, welches das gleichnamige Grundverhalten beinhaltet, hat lediglich die Vorbedingung, daß der Ball von irgendeinem Spieler gesehen wird. *ObserveBall* ist ein relativ passives Verhalten, das nur dann ausgeführt werden soll, wenn gerade nichts Anderes zu tun ist. Da außerdem der aktive Spieler immer etwas Besseres zu tun haben sollte, besitzt dieses Modul nur den Effekt *cooperate* mit (geringer) Stärke 0.2

SearchBall dient dazu, den Ball mit Hilfe des entsprechenden Grundverhaltens zu suchen, falls ihn weder der Spieler selbst noch irgendein Mitspieler sehen kann. Genau dann ist die einzige Vorbedingung *not ball_present* erfüllt. Sowohl der aktive Spieler wie auch seine Mitspieler sollen nach dem Ball suchen. Deshalb wurde *SearchBall* mit beiden Zielen verknüpft. Nur die Kompetenzmodule *GotoStrategicPosition* und *GotoSupportPosition* (s.u.) sind ausführbar falls *not ball_present* gilt (sämtliche Propositionen, die irgendetwas mit dem Ball zu tun haben, werden dann *falsch*). Daher mußte die Effektstärke auf *cooperate* mit 0.6 etwas höher gewählt werden als diejenige dieser Module. Die Effektstärke auf *soccergoal* beträgt 0.5.

Ballbesitz

Um in Ballbesitz zu gelangen, setzen die Roboter das Grundverhalten *Approach-Ball* ein, dem zwei Kompetenzmodule **GetBall** entsprechen. Sich dem Ball auf direktem Wege anzunähern ist dann sinnvoll, wenn er sich schon in unmittelbarer Nähe des Spielers befindet (*getball_ball_close*) oder wenn der direkte Weg zum Ball frei ist, die Distanz zum Ball nicht zu groß ist und die Roboterposition einen guten Winkel zur Ballposition aufweist. Der momentan beste Winkel zur Ballposition ergibt sich dabei aus der dynamischen *Aktiv-Position*, deren Berechnung in Abschnitt 5.5.3 beschrieben wird. Auch die Position der Gegenspieler wird berücksichtigt: Je näher der nächste Gegenspieler im Vergleich zum aktiven Spieler zum Ball steht, desto größer darf dessen Abweichung vom optimalen Winkel für ein *Approach-Ball* sein. Beide Kompetenzmodule haben den Effekt *have_ball* mit Stärke 0,9.

Sind die obigen Vorbedingungen nicht erfüllt, so soll der Spieler mittels *Goto-Position* (d.h. mit Hilfe des Pfadplaners) die aktuelle *Active*-Position in der Nähe des Balls anfahren. Das entsprechende Kompetenzmodul **GotoBall** setzt ansonsten lediglich voraus, daß die Ballposition bekannt ist (*ball_present*). Die

Effekte von *GotoBall* entsprechen den Vorbedingungen des zweiten *GetBall*-Moduls, jeweils mit Effektstärke 0.9.

Die Kompetenzmodule dieser Gruppe sind die einzigen, die keinen direkten Effekt auf eines der Ziele haben. Die Effektverknüpfungen wurden so gewählt, daß sich im Idealfall die Verhaltenssequenz [*GotoBall* → *GetBall* → *Spiel mit Ball*] ergibt. Wie im letzten Abschnitt erwähnt, wird in solchen Situationen, in denen das *soccer_goal* relevant ist, die Proposition *have_ball* aber noch nicht erfüllt, letztere zu einem Unterziel. Über das nicht ausführbare Kompetenzmodul *ShootGoal* fließt Aktivierung zu den beiden *GetBall*-Modulen, welche das Unterziel *have_ball wahr* machen können. Ist *GetBall* ebenfalls nicht ausführbar z.B., weil die Distanz zum Ball noch zu groß ist, so wird jede unerfüllte Vorbedingungsproposition zu einem weiteren Unterziel des Netzwerks, über das Aktivierung an *GotoBall* weitergegeben wird. Wird *GotoBall* zur Ausführung ausgewählt, so verbessert sich im Idealfall der Wahrheitswert der Vorbedingung von *GetBall*, weil sich der Spieler zu einer optimalen Position in der Nähe des Balls bewegt. Sobald sich der Spieler in einer guten Position für ein *Approach-Ball* befindet, wird das Kompetenzmodul *GetBall* ausführbar, *GotoBall* erhält keine Aktivierung mehr und als nächstes wird *GetBall* ausgeführt (wenn nicht z.B. auch die Voraussetzungen für einen *Bump-Shoot* erfüllt sind, s.u.). Ist auch das direkte Heranfahren an den Ball erfolgreich, so wird *have_ball wahr*, d.h. *GetBall* erhält seinerseits keine Aktivierung mehr, und stattdessen wird ein geeignetes anderes Kompetenzmodul ausgewählt, um den Ball zu spielen. Man beachte, wie sich hier durch die Möglichkeit, Kompetenzmodule zu verknüpfen, und die Eigenschaften der Aktivierungsweitergabe das gleiche Verhalten ergibt, wie es bei der letztjährigen Aktionsauswahl durch Kontrollfunktionen realisiert wurde.

Spiel mit Ball

Den folgenden 5 Kompetenzmodulen ist gemeinsam, daß sie nur dann ausführbar sind, wenn sich der Spieler in Ballbesitz befindet. Sie besitzen außerdem alle den Effekt *soccergoal*. Die gewählten Effektstärken (*ShootGoal* 0.8, *MoveShoot-Feint* 0.7, *DribbleBall* 0.6, *TurnBall* 0.2, *WaitAndBlock* 0.2 und *TurnAwayBall* 0.1) spiegeln hier noch am ehesten die Erwahrtungswahrscheinlichkeit eines Torerfolgs wieder: Betrachtet man Situationen, in denen sich der Spieler mittels *DribbleBall* in Richtung gegnerisches Tor bewegt, so wird daraus im Mittel weniger häufig ein Tor resultieren als aus direkten Torschüssen *ShootGoal*.

Das Kompetenzmodul **ShootGoal** realisiert den direkten Torschuß mit Hilfe des Grundverhaltens *Shoot-to-Pos*. Vorbedingungen sind, daß sich der Spieler in Schußdistanz zum gegnerischen Tor befindet, daß eine freie Stelle im gegnerischen Tor existiert (*goal_free*), daß vor dem Spieler genug Platz ist, um sich zu dieser freien Stelle hin auszurichten, und daß die Winkeldifferenz zwischen aktueller und angestrebter Ausrichtung nicht zu groß ist. *ShootGoal* kann nicht ausgeführt werden, falls direkt vorher eines der Kompetenzmodule *TurnAway-*

Ball, *BumptShootDef*, *BumpShootOff*, *MoveShootFeint* oder *FlipperShoot* war (*shootgoal_excl_lastact*). *Aktiv* bedeutet, daß keines dieser Module im letzten *Saphira*-Zyklus[4] zur Ausführung ausgewählt wurde. Zum Konzept dieser exklusiven Vorgängeraktionen (*exclusive lastactions*) siehe Abschnitt 5.4.3. Die Position, zu der das Grundverhalten schießen soll, ergibt sich aus der Berechnung von *goal_free*. Mit Hilfe des Potentialfelds $f_{goalway}$ (siehe Seite 21) wird eine geeignete Zielzelle auf der gegnerischen Torlinie bestimmt und ihr Mittelpunkt dem Grundverhalten übergeben.

MoveShootFeint basiert auf der in Kapitel 3.3 angeführten Verbundaktion (*compound action*) gleichen Namens. Diese steuert mit Hilfe des Grundverhaltens *Play-Ball* direkt das gegnerische Tor an. Deshalb ist dieses Kompetenzmodul ausführbar, falls der Spieler reichlich Platz vor sich hat, seine Ausrichtung einen guten Winkel zum Tor aufweist und er nicht zu weit entfernt vom gegnerischen Tor ist. Darüber hinaus darf sich kein Gegenspieler innerhalb des dreieckförmigen Bereichs befinden, der sich aus der Verlängerung (Roboterposition $\rightarrow$ linker Torpfosten) auf der einen und (Roboterposition $\rightarrow$ rechter Torpfosten) auf der anderen Seite bis zur gegnerischen Strafraumgrenze ergibt (*goalarea_free*). *MoveShootFeint* besitzt die exklusiven Vorgängeraktionen *ShootGoal*, *TurnAwayBall*, *BumpShootOff* und *BumpShootDef*.

Das Modul **DribbleBall** benutzt die zweite der zur Verfügung stehenden zusammengesetzten Aktionen (*Dribble-KickAway-Ball*). Ein „Dribbling"ist möglich, falls der Spieler nur etwas Platz vor sich hat und seine (absoluter) Orientierungswinkel einen bestimmten Schwellwert nicht übersteigt. Das letzte ausgeführte Kompetenzmodul darf nicht *ShootGoal*, *MoveShootFeint*, *TurnAwayBall*, *BumpShootOff* oder *BumpShootDef* sein.

Zeigt die Ausrichtung des Spielers im Ballbesitz eher zum eigenen als zum gegnerischen Tor, so kann er das Grundverhalten *Turn-Ball* einsetzen. Da die Drehung mit Ball ein wenig Raum beansprucht, beinhaltet die Vorbedingung *turnball_angles_clear* des Kompetenzmoduls **TurnBall** u.a., daß der Spieler sich dabei nicht zu nahe an der Bande, direkt vor dem eigenen Tor oder in unmittelbarer Nähe des Gegners befinden soll. Darüber hinaus darf nicht *ShootGoal*, *MoveShootFeint*, *Dribble-Ball*, *TurnAwayBall*, *BumpShootOff* oder *BumpShootDef* zuletzt aktiv gewesen sein.

Die Ausführbarkeit des Kompetenzmoduls **TurnAwayBall** ist abgesehen vom Ballbesitz lediglich beschränkt durch die exklusiven Vorgängermodule *ShootGoal*, *MoveShootFeint*, *TurnAwayBall*, *BumpShootOff* und *BumpShootDef*. Es hat die schwächste Vorbedingung der bisher vorgestellten Ballspiel-Module, aber auch die niedrigste Effektstärke auf *soccergoal*. Es wird daher nur dann ausgeführt, wenn keines der anderen Module anwendbar ist, beispielsweise wenn

[4] Gemeint ist hier der in Kapitel 3.3 beschriebene Gesamt-Zyklus, den die Spielerarchitektur durchläuft. In jedem Durchlauf wird genau eine Aktion ausgewählt und ausgeführt. Im Gegensatz dazu braucht der Aktionsauswahl-Zyklus in Kapitel 5.3.3 im allgemeinen mehrere Durchläufe, bis ein Kompetenzmodul den Schwellwert θ überschreitet und ausgewählt wird.

der Ball sehr nahe an der Bande liegt.

Kompetenzmodul **WaitAndBlock**: Liegt eine Block-Situation vor, d.h. sowohl der Gegner als auch der Spieler sind in Ballbesitz, und befindet sich der Ball in der Nähe des eigenen Tors, dann ist es günstig, einfach abzuwarten und den Gegner zu blockieren, anstatt zu versuchen, z.B. mit *TurnAwayBall* oder *TurnBall* dem Gegner den Ball abzunehmen. Da nach etwa 10 Sekunden Blockierung der Schiedsrichter eingreifen wird und den Ball an eine nahegelegene freie Stelle plazieren, kann ein Mitspieler die Block-Situation erkennen und sich entsprechend postieren, um die eigene Mannschaft wieder in Ballbesitz zu bringen. Ausgeführt wird dabei das Grundverhalten *Observe-Ball*.

BumpShoot

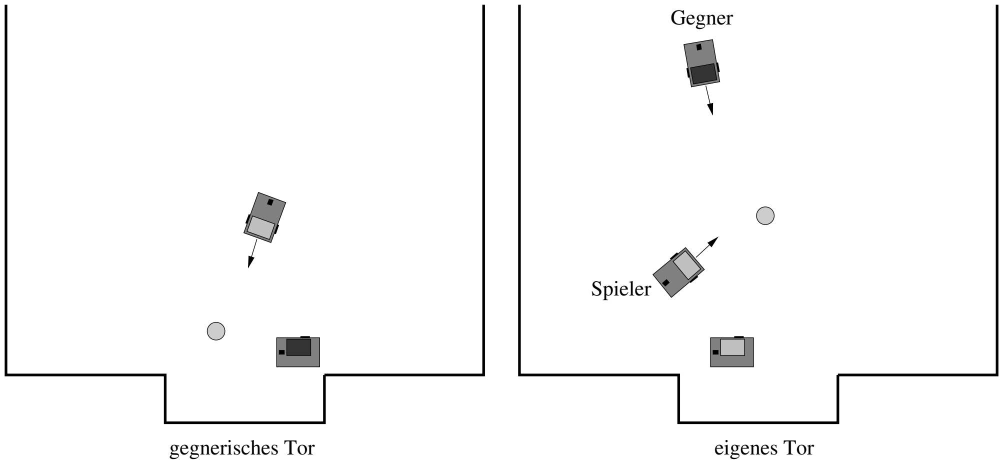

Abbildung 5.3: Zwei Situationen für einen *Bump-Shoot*.

Die beiden folgenden Kompetenzmodule spielen ebenfalls den Ball. Im Gegensatz zu den gerade beschriebenen haben sie jedoch die Vorbedingung *not have_hall*. In bestimmten Situationen, wenn der Spieler zwar noch nicht in Ballbesitz ist, aber schon günstig zum Ball steht, soll der Ball mit dem Grundverhalten *Bump-Shoot* „weggeschlagen " werden. „Günstig zum Ball heißt dabei, daß der direkte Weg zum Ball frei ist und daß die Distanz zum Ball nicht zu groß ist.
Abbildung 5.3 zeigt die zwei typischen Situationen für einen *Bump-Shoot*:
Liegt der Ball im gegnerischen Strafraum und befindet sich der Spieler in gutem Winkel zum gegnerischen Tor, dann ist die Gelegenheit für **BumpShootOff** (Off = offensiv). **BumpShootDef** dagegen kann den Ball klären, falls dieser sich in der eigenen Hälfte befindet, aber nicht zu nahe an der Bande liegt, und ein Gegner in Ballnähe ist. Die Vorbedingung *good_bumpdef_angle* drückt aus, daß der Ball natürlich nur in Richtung gegnerisches Tor geklärt werden soll. Da defensive *Bump-Shoots* leicht zu einem etwas rüden Spiel führen, existiert ein globales Flag, um diese Aktion generell zu verbieten. Der Wahrheitswert von

allow_fast_defense spiegelt den Status dieses Flags wieder.
Beide Module haben den Effekt *soccergoal* mit Effektstärke 0.9. Das bedeutet: In den speziellen Situationen, in denen sie ausführbar sind, sollen sie allen anderen Modulen (Ausnahme: *FlipperShoot*, siehe unten) vorgezogen werden.

Paßspiel

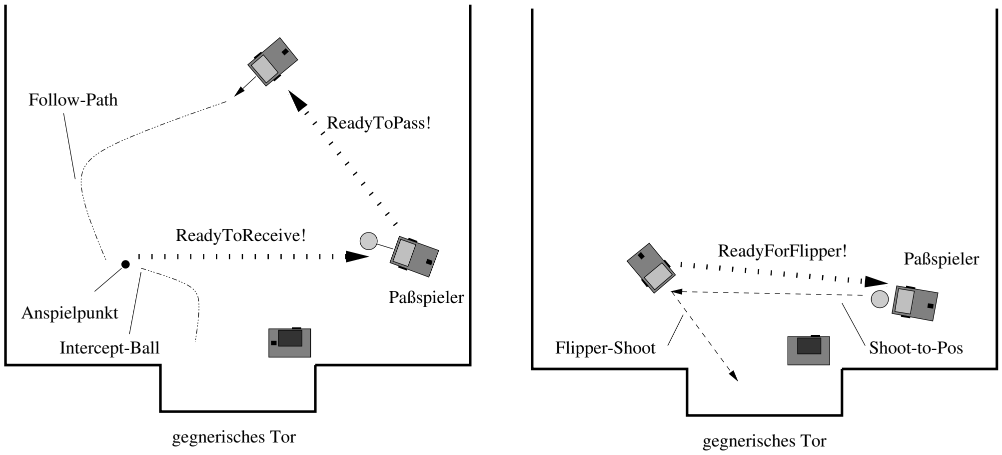

Abbildung 5.4: Das alte und neue Paßspiel.

Im Verlauf des *RoboCup'99* zeigten die Spieler des CS Freiburg zum ersten Mal überhaupt für die *Middle Size League* ein Paßspiel vor dem gegnerischen Tor. Die Kooperation der Spieler verlief dabei wie folgt (siehe Abbildung 5.4, links): Der Spieler im Ballbesitz befindet sich neben dem gegnerischen Tor und erkennt, daß ein direkter Torschuß nicht aussichtsreich ist. Der direkte Weg zu einer vordefinierten Paßposition ist jedoch frei. Er sendet deshalb eine Nachricht (*ReadyToPass*) an den Mitspieler, der am günstigsten zur Paßposition steht, woraufhin dieser sich dorthin begibt, sich zum Paßspieler orientiert und eine Bestätigungsnachricht sendet (*ReadyToReceive*). Jetzt wird der Paß gespielt und der Empfänger versucht, den Ball unter Kontrolle zu bringen und einen Torschuß auszuführen.
Problematisch bei dieser Vorgehensweise ist, daß es einige Sekunden dauern kann, bis der Paßempfänger den Anspielpunkt erreicht hat. Je besser die gegnerische Mannschaft spielt, desto unwahrscheinlicher ist es, daß der Spieler im Ballbesitz in dieser Zeit nicht angegriffen wird. Darüber hinaus machten es die stark gestiegene Schußkraft der Roboter sowie das Fehlen eines geeigneten Grundverhaltens zur Ballannahme notwendig, daß Paßkonzept zu ändern (siehe Abbildung 5.4, rechts):
Der Spieler mit der Rolle *Support* geht jetzt im Rahmen seines dynamischen Stellungsspiels selbstständig zum (immer noch) festgelegten Anspielpunkt und kann sich dort zum Paß „anbieten“, indem Nachrichten *ReadyForFlipper* an den aktiven Mitspieler gesendet werden. Dazu dient das Kompetenzmodul **FlipperShoot**, welches das gleichnamige Grundverhalten benutzt. Vorbedingungen

sind hier, daß sich der Spieler am Anspielpunkt befindet und daß er die Rolle *Support* hat bzw. hatte, als das Grundverhalten gestartet wurde. Dies ist nötig, weil der Paßempfänger normalerweise die Rolle *Active* bekommt, sobald der Ball in seiner Nähe ist, das Grundverhalten *Flipper-Shoot* dann aber noch nicht beendet werden darf. Außerdem muß es aus seiner Sicht noch möglich sein, einen Paß anzunehmen. Letztere Proposition wird u.a. falsch, falls der Gegner in Ballbesitz ist oder der Paß zwar gespielt wurde, den Paßempfänger aber verfehlt hat.

Dem aktiven Spieler in Ballbesitz steht das Kompetenzmodul **FlipperPass** zur Verfügung. Es ist ausführbar, falls ein Mitspieler sich zum Paß anbietet und die momentane Orientierung, Position und Geschwindigkeit einen Schuß zum Anspielpunkt zulassen. Exklusive Vorgängeraktionen sind *ShootGoal*, *MoveShootFeint*, *Dribble-Ball*, *TurnAwayBall*, *BumpShootOff* und *BumpShootDef*. Die Vorbedingung *not goal_free* bewirkt, daß nur gepaßt wird, falls ein direkter Torschuß geringe Erfolgsaussichten hat. Das Abspiel erfolgt dann mit Hilfe des Grundverhaltens *Shoot-to-Pos*.

Die Proposition *goal_free* besitzt einen echt kontinuierlichen Wertebereich, so daß sich die Vorbedingungen der Module *ShootGoal* und *FlipperPass* nicht gegenseitig ausschließen. Für den Effekt *soccergoal* des letzteren Kompetenzmoduls wurde im Vergleich zu *ShootGoal* eine etwas höhere Effektstärke (0.85) gewählt, um dem Paßspiel in solchen Situationen, in denen beide Module die gleiche Ausführbarkeit besitzen, einen leichten Vorteil zu geben.
Wegen des möglichen Rollenwechsels (s.o.) mußte *FlipperShoot* mit beiden Zielen verknüpft werden. Für den kooperierenden Spieler ist ein *Flipper-Shoot* das Beste, was er machen kann (Effektstärke 0.8). Der aktiv gewordene Spieler soll die begonnene Aktion auf jeden Fall weiterführen, solange sie ausführbar ist (Effektstärke 1.0).

Stellungsspiel

Falls ein Spieler gerade nicht die Rolle des aktiven Spielers hat, so hat er vor allem die Aufgabe, eine gute Position einzunehmen. Je nach eingenommener Rolle existieren zwei Kompetenzmodule **GotoSupportPosition** und **GotoStrategicPosition**, die beide das Grundverhalten *Go-to-Position* benutzen. Die jeweils einzige Vorbedingung *not in_good_support_position* bzw. *not in_good_strategic_position* bewirkt, daß sich der Spieler nur dann zu der momentan für ihn besten Position bewegt, falls er sich nicht innerhalb eines *Erfolgsradius* um diese Position befindet. Durch Variieren des Erfolgsradius kann also kontrolliert werden, wie „agil“ das Stellungsspiel der nicht-aktiven Spieler ist. Die Vorbedingungen der beiden Kompetenzmodule schließen sich gegenseitig aus, weshalb beide mit Stärke 0.5 mit dem Ziel *cooperate* verknüpft sind.

5.4.3 Diskussion

Wie sich bei der Erstellung des oben beschriebenen Verhaltensnetzwerks gezeigt hat, sind innerhalb des bestehenden Systems die Möglichkeiten der Ausgestaltung einer Komponente zur Aktionsauswahl relativ beschränkt. Einige dieser Beschränkungen sollen in den folgenden Abschnitten näher betrachtet werden. Außerdem wird kurz auf *Reflexe* sowie auf die konkrete Implementierung der Erweiterten Verhaltensnetzwerke eingegangen.

Zusammengesetzte Aktionen

Es ist offensichtlich, daß sich die Gestaltung einer Strategiekomponente daran orientieren muß, welche Grundverhalten zur Verfügung stehen, wie sie aufgebaut sind und was sie leisten. Einige Grundverhalten z.B. *Shoot-To-Pos* bilden relativ kleine Einheiten und sind darüber hinaus parametrisierbar. Andere jedoch sind mächtiger und besitzen eine komplexere innere Struktur. Ein Beispiel hierfür ist die zusammengesetzte Aktion *Move-Shoot-Feint* (vgl. S. 23). Mächtige, zusammengesetzte Grundverhalten sind für die Aktionsauswahl nicht so einfach zu handhaben. Normalerweise soll der direkte Torschuß mit *ShootGoal* einem *MoveShootFeint* vorgezogen werden, falls beide ausführbar sind. Daher besitzt das Modul *ShootGoal* eine höhere Effektstärke. Während der Ausführung von *MoveShootFeint* ergeben sich häufig Situationen, in denen auch ein direkter Torschuß möglich wäre, hier soll *MoveShootFeint* jedoch nicht durch *ShootGoal* abgelöst werden, weil es ja selbst einen Torschuß einleitet.
Im letzten Jahr wurde dieses Problem durch *Persistenzwerte* gelöst, die den Aktivierungswert eines Aktionsmoduls bestimmten, während es ausgeführt wurde und die höher waren als die Aktivierungswerte all derer Aktionsmodule, die ebenfalls ausführbar waren aber nicht ausgeführt werden sollten. In das diesjährige Verhaltensnetzwerk wurden zu diesem Zweck Vorbedingungspropositionen eingeführt, die *falsch* werden, falls zuletzt eine für das zugehörige Kompetenzmodul *exklusive Vorgängeraktion* ausgeführt wurde. Beispiele für weitere zusammengesetzte Aktionen, welche die Definition derartiger Propositionen erfordern, sind *Dribble-KickAway-Ball* und *Bump-Shoot.*

Persistenz

Beim Abwägen zwischen Persistenz des Verhaltens auf der einen und Reaktivität auf der anderen Seite haben die praktischen Erfahrungen mit dem Sstem gezeigt, daß es in vielen Fällen zu besseren Ergebnissen führt, wenn mehr Wert auf Verhaltenspersistenz gelegt wird. Der Grund dafür sind zum einen Beschränkungen, die sich aus der Mechanik der Roboter ergeben: Bewegt sich der Spieler z.B. bereits mit hoher Geschwindigkeit geradeaus und ist in Ballbesitz, so ist das Fahren eines relativ engen Bogens (z.B. um ein Hindernis) nicht mehr möglich, ohne dabei den Ball zu verlieren. Weitere Beschränkungen sind dadurch gegeben, daß die geschätzte Ballposition oft starken, nicht

der realen Ballbewegung entsprechenden Schwankungen unterliegt, insbesondere dann, wenn der Roboter sich bewegt. Daher ist es oft günstig, eine Aktion auch dann weiterzuführen, wenn Propositionen der Vorbedingung des entsprechenden Kompetenzmoduls nicht mehr oder nicht mehr ganz erfüllt sind.
Im obigen Verhaltensnetzwerk wird Verhaltenspersistenz zumeinen durch das Ausschließen bestimmter Aktionsfolgen mit Hilfe der exklusiven Vorgängeraktionen erreicht (z.B. *MoveShootFeint* $\rightarrow$ DribbleBall). Zum anderen besitzen die meisten kompetenzmodul-spezifischen Propositionen einen besseren Wert (prüfen also eine schwächere Bedingung), falls das zugehörige Kompetenzmodul gerade aktiv ist. Der Agent bezieht also seinen internen Zustand (d.h. welche Aktion er gerade ausführt) mit in die Auswertung der Umgebung ein. Die Aktionsmodule der letztjährigen Implementierung besaßen zum gleichen Zweck den *Persistenzwert* der Aktivierungsfunktion und eine Menge von *Invarianten* (siehe Abschnitt 5.2).

situationsabhängige Ziele

Das oben beschriebene Netzwerk enthält zwei Ziele, von denen aber in jeder Situation — je nach Rolle des Spielers — nur genau eines relevant ist. Die Spieler der Simulationsmannschaft von Dorer hatten dagegen einen echten Zielkonflikt zu lösen: Sie mußten sich jeweils entscheiden, ob es besser war, sich auszuruhen und Ausdauer zu sparen bzw. zu regenerieren, oder ob die Spielsituation es unbedingt erforderte, z.B. zum Ball zu rennen. Für den CS Freiburg spielt Ausdauerproblematik zur Zeit jedoch keine Rolle (mehr), da die Batterien der Roboter genug Energie für ein ganzes Spiel haben.
An einem Beispiel soll gezeigt werden, wie mit situationsabhängigen komplexere taktische Verhaltensweisen relativ elegant ausgedrückt werden können. Dazu sollen Situationen betrachtet werden, in denen sich ein Spieler in der Nähe des gegnerischen Tors befindet und sowohl ein Torschuß, als auch ein Paß zu einem Mitspieler möglich sind. Verfolgt der Spieler das Ziel, ein Tor zu schießen, so sind normalerweise die Erfolgsaussichten eines direkten Torschusses höher als die eines Abspiels. Im Verhaltensnetzwerk entspricht dies einer höheren Effektstärke des Kompetenzmoduls *ShootGoal* auf das Ziel *soccergoal* als *FlipperPass* (siehe Abbildung 5.5). Zusätzlich soll als zweite Zielvorgabe gelten, ein möglichst ansprechendes Spiel zu zeigen (*play_nice*), allerdings mit geringerer Wichtigkeit. Da ein Paßspiel in höherem Maße zu einer schöneren Spielweise beiträgt als ein direkter Torschuß, ist diesmal die Verknüpfung von *FlipperPass* mit dem Ziel *play_nice* stärker gewichtet als die entsprechende Verknüpfung des Moduls *ShootGoal*. Bei der Verwendung nur statischer Ziele dominiert stets der Wunsch, Tore zu erzielen, aber nur leicht. Die Aktivierung, die das Modul *ShootGoal* erhält, beträgt dann nach Gleichung 5.1 und 5.6 (ohne Berücksichtigung von β) $\gamma \cdot 1.0 \cdot 0.9 + \gamma \cdot 0.8 \cdot 0.5 = \gamma \cdot 1.3$, im Vergleich zu $\gamma \cdot 0.8 \cdot 0.9 + \gamma \cdot 1.0 \cdot 0.4 = \gamma \cdot 1.12$ bei *FlipperPass*. Ein Paß würde daher immer dann gespielt, falls seine Ausführbarkeit nur etwas höher ist als die eines Torschusses. Gerade zu Beginn eines Spiels falls die eigene Mannschaft zurückliegt, ist dies vermutlich nicht erwünscht. Natürlich kann die Wichtigkeit des Ziels *play_nice* entsprechend gesenkt wer-

den, dann käme der Paß jedoch eventuell insgesamt zu selten zum Zuge. Eine gute Spielstrategie könnte sein, um so eher mal einen Paß zu spielen, je klarer die eigene Mannschaft führt, bei knappen Spielständen dagegen häufiger direkt aufs Tor zu schießen, auch wenn vielleicht ein Paß eine höhere Ausführbarkeit besitzt. Dies wird durch Hinzufügen einer Relevanzbedingung zu beiden Zielen erreicht (siehe Abbildung 5.5). Die Proposition *own_team_leading* bildet dabei den aktuellen Spielstand auf ein kontinuierliches (subjektives) Maß für den Vorsprung der eigenen Mannschaft ab. Die Skalierung der Propositionwurde so gewählt (bzw. durch einfache Gleichungsumformung berechnet), daß bei gleicher Ausführbarkeit ab vier Toren Vorsprung der eigenen Mannschaft das Paßspiel dem Torschuß vorgezogen wird. Die Abbildungen 5.6 und 5.7 veranschaulichen das resultierende Verhalten[5]: Das Kompetenzmodul *ShootGoal* hat hier eine etwas schlechtere Ausführbarkeit (0.5) als *FlipperPass* (0.8). Beim Spielstand von 0:0 wird jedoch trotzdem ein Torschuß probiert, da das Ziel *play_nice* hier noch keine Relevanz für den Spieler hat. Falls die eigene Mannschaft jedoch mit drei Toren in Führung liegt, ist die Motivation, ein schönes Spiel zu zeigen, entsprechend gestiegen. Noch mehr Tore zu erzielen ist für den Spieler dagegen nicht mehr so relevant. Daher spielt er diesmal in der gleichen Situation einen Paß.

Parameter

Nur in sehr einfachen Fällen wie dem obigen Beispiel oder in eher eingeschränkten Netzwerken wie dem Verhaltensnetz für das diesjährige *RoboCup*-Turnier können die Parameter der Aktionsauswahl (globale Netzparameter, Effektstärken, Skalierung der Propositionen) analytisch berechnet bzw. intuitiv oder anhand relativ einfacher Prinzipien bestimmt werden. Bei der Verwendung komplexerer Netzwerke ist die Parameterbestimmung jedoch in der Regel nur noch empirisch möglich, eine in der Regel schwierige und zeitaufwendige Arbeit. Wünschenswert wäre daher der Einsatz von Lernverfahren. *Maes* beschreibt beispielsweise ein Verfahren zur automatischen Anpassung der Netzparameter[1991]. Benutzt wird dabei ein Meta-Verhaltensnetzwerk, dessen Aktionen die globalen Parameter des eigentlichen Netzwerks beeinflussen. auch Verfahren zum Lernen von Effektstärken wurden vorgeschlagen[Maes, 1992]. Generell macht die Tatsache, daß es sich bei den Spielern des *CS Freiburg* um reale Roboter handelt (im Gegensatz zu den Agenten der Simulationsliga), den Einsatz automatischer Verfahren wie auch die empirische Parameterbestimmung besonders aufwendig und zeitraubend (z.B. weil immer wieder der Ball an die selbe Stelle gelegt werden muß).

Reflexe

Das Aktionsmodul *ActionFreeFromStall* der letztjährigen Aktionsauswahlkomponente besitzt keine Entsprechung im Verhaltensnetzwerk. *ActionFreeFrom-*

[5]Gegner sind rot markiert, der potentielle Paßempfänger grün.

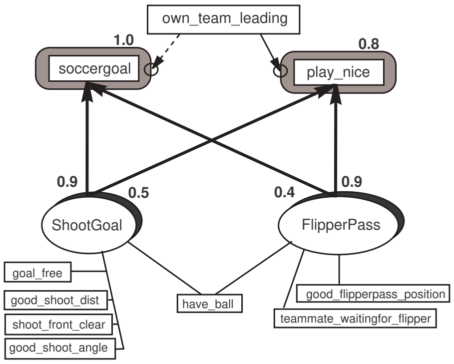

Abbildung 5.5: Beispielnetzwerk.

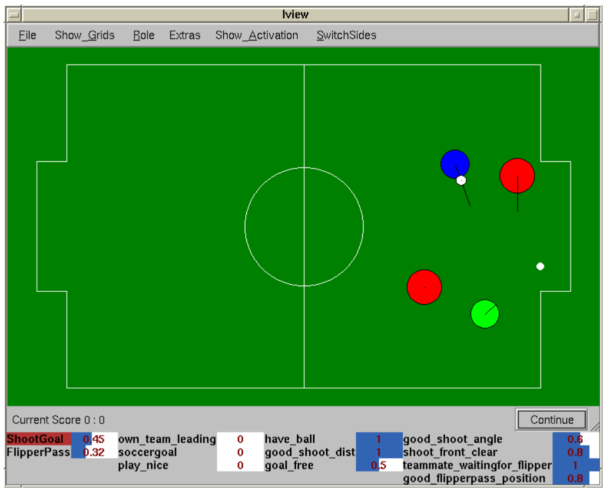

Abbildung 5.6: Spielstand 0:0. Der Torschuß wird vorgezogen.

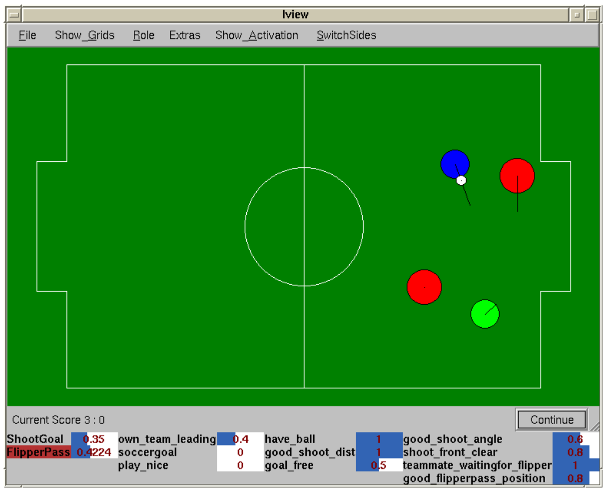

Abbildung 5.7: Spielstand 3:0. Der Paß wird vorgezogen.

Stall diente dazu, das Grundverhalten *Free-From-Stall unbedingt* auszuführen, falls der Roboter festhing und nicht in Ballbesitz war. Alle anderen Aktionsmodule enthielten dazu die Invariante *not RobotStalls*. Für die diesjährige Realisierung schien es eleganter, für derart spezielle Situations-Aktions-Regeln sogenannte Reflexe vorzusehen. Reflexe haben in vorgegebenen Situationen eine direkte Ausführung eines bestimmten Grundverhaltens ohne Einbeziehung des Verhaltensnetzwerks zur Folge.
Neben dem Reflex zur Behandlung von *Stall*-Situationen existiert noch ein zweiter zur Einhaltung der „10-Sekunden-Regel“ (siehe Kapitel 3.1). Sobald der Spieler merkt, daß er sich zu lange im eigenen oder gegnerischen Strafraum befindet, ruft er das Grundverhalten *Go-To-Pos* mit dem Feldmittelpunkt als Zielpunkt auf. Der Reflex bleibt solange aktiv, bis der Roboter sich sicher außerhalb des Strafraums befindet. Anschließend übernimmt wieder das Verhaltensnetzwerk die Aktionsauswahl.

Implementierung

Bei der Implementierung der letztjährigen Strategiekomponente wurden jeweils sämtliche Auswertungen für Prädikate der Vorbedingung eines Aktionsmoduls in einer sog. Vorbedingungs- bzw. Invarianzfunktion (deren Funktionswert entsprechend *wahr* oder *falsch* war) zusammengefaßt. Dies machte die Vorbedingungen komplexerer Aktionsmodule wie z.B. *ActionBumpShoot* schnell relativ unübersichtlich und führte auch zu unnötigen Codewiederholungen und Mehrfachauswertungen. Daher wurden die Propositionen des Weltzustands als eigenständige Objekte gekapselt. Auch sonst wurde Wert auf eine möglichst ausgeprägte Modularisierung der einzelnen Komponenten gelegt.
Die Spezifikation eines Verhaltensnetzwerks erfolgt komfortabel und flexibel als Textdatei (siehe Anhang C). Bei Systemstart wird die gewünschte Datei eingelesen und intern das Netzwerk mit seinen Verknüpfungen aufgebaut. Nur in der Netzspezifikation genannte Propositionen werden ausgewertet, alle anderen stehen aber weiterhin zur Verfügung und können — falls erforderlich — ohne Neukompilierung des Systems in die Verhaltensauswahl eingebunden werden.
Der Nachteil dieser Strukturierung ist der höhere Aufwand bei der Implementierung neuer Propositionen. Allerdings kann man davon ausgehen, daß dies mehr als ausgeglichen wird durch kürzere Einarbeitungs- und Wartungszeiten.

5.5 Kooperatives Zusammenspiel

In Abschnitt 5.4.2 wurde ein Beispiel für ein kooperatives Zusammenspiel zweier Spieler vorgestellt: Das Paßspiel. In diesem Abschnitt soll es um das kooperative Mannschaftsspiel gehen. Zwei grundlegende Probleme sind dabei zu lösen:

1. Es sollen nicht alle Spieler gleichzeitig zum Ball gehen und sich dabei gegenseitig behindern.

2. Die Spieler sollen sich so positionieren, daß sich eine günstige Raumaufteilung ergibt.

Eine gute Raumaufteilung hat mehrere Vorteile: Die eigene Mannschaft ist schnell am Ball, sie macht „die Räume eng" für den Gegner, immer mindestens ein Spieler sieht den Ball und die Spieler stehen einander nicht im Weg.

5.5.1 Ausgangspunkt

Bei der Version des CS Freiburg, die 1998 in Paris antrat[Weigel, 1998], wurden beide Probleme durch den Einsatz eines festgelegten Kompetenzbereichs pro Spieler gelöst. Ein Spieler wurde aktiv, sobald sich der Ball in seinem Kompetenzbereich befand und durfte diesen mit Ball auch verlassen. Für letzteren Fall sendete der Spieler eine spezielle Nachricht *ClearOut*, die seine Mitspieler veranlaßt, ihm aus dem Weg zu gehen. Ansonsten bewegte er sich jedoch zu seiner vordefinierten Grundposition, um von dort aus das Spielgeschehen zu beobachten. Es zeigte sich, daß durch diese starre Aufteilung zwar erfolgreich vermieden werden konnte, daß die Roboter sich gegenseitig behindern, daß jedoch oftmals der Spielfluß darunter litt.

Deshalb verzichtet die letztjährige Strategiekomponente auf Kompetenzbereiche[6] [Reetz, 1999]. Stattdessen wird jedem Spieler eine Rolle zugewiesen, und zwar entweder Verteidiger oder Stürmer. Die feste Mannschaftsformation besteht aus zwei Verteidigern (links und rechts) und einem Stürmer. Für jede dieser Rollen existieren eine Anzahl fester Grundpositionen, aus denen der Spieler je nach Spielsituation (d.h. je nachdem in welchen Drittel des Feldes sich der Ball gerade befindet) eine auswählt. Falls die Spielsituation es erlaubt, veranlaßt einer der beiden Verteidiger den anderen durch Versenden einer Nachricht *GoForward*, ins Mittelfeld aufzurücken, während er selbst eine Liberoposition einnimmt. Der Stürmer reagiert auf diese Absprache, indem er sich zu der vom Mittelfeldspieler aus gesehenen entgegengesetzten Seite des Spielfelds bewegt (siehe Abbildung 5.8).

Um zu entscheiden, welcher der Spieler zum Ball geht, enthält die Vorbedingung des Aktionsmoduls *ActionGetBall* Prädikate, die angeben, ob sich ein Mitspieler in einer besseren Position zum Ball befindet und ob bereits ein Mitspieler signalisiert hat, zum Ball zu gehen. Beide Prädikate müssen *falsch* sein, damit das Modul ausführbar ist. Ist das der Fall, dann versendet der Spieler eine Nachricht *GotoBall*, um seinen Mitspielern zu signalisieren, daß der zum Ball geht. Die *ClearOut* Nachricht wurde beibehalten und immer dann gesendet, falls sich ein Spieler aufs Tor zubewegte.

[6] Auf Kompetenzbereiche wurde und wird nur dann zurückgegriffen, falls die Funkverbindung der Spieler ausfällt. Da die Spieler dann nicht mehr zwischen Gegenspielern und Mitspielern unterscheiden können, kann nur auf diese Weise ein sehr schlechtes Mannschaftsspiel verhindert werden.

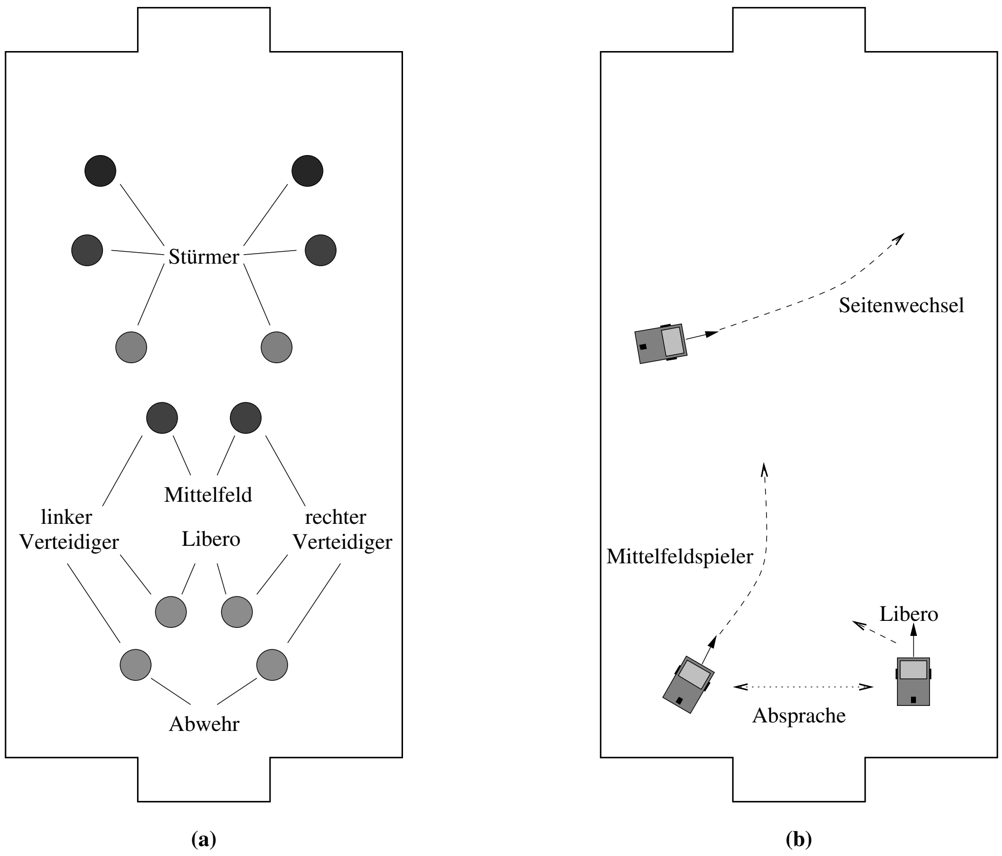

Abbildung 5.8: Die Mannschaftstaktik 99.

Insgesamt erreicht Reetz ein wesentlich flexibleres Spiel, bei dem nicht mehr offensichtliche Chancen aufgrund starrer Regeln ausgelassen werden. Ein Rollentausch zwischen Stürmer und Verteidiger, beispielsweise falls der Verteidiger gerade mittels *Move-Shoot-Feint* einen Angriff versucht hat und sich in der Nähe des gegnerischen Tores befindet, ist jedoch nicht vorgesehen. Eine weitere Verbesserungmöglichkeit ist die Aufgabe der festen Grundpositionen, die eine nur grobe Klassifikation der Spielsituation erlauben, zugunsten dynamischer, d.h. jeweils anhand der aktuellen Spielsituation *berechneter* Positionen.

5.5.2 Dynamische Rollen

Das diesjährige Konzept erlaubt es den Spielern, Rollen dynamisch einzunehmen und abzugeben. Zunächst wurden dazu drei Rollen definiert[7] (vgl. [Castelpietra *et al.*, 2000]):

- *Active*: Der aktive Spieler versucht stets, den Ball zu spielen oder in Ballbesitz zu kommen. Verglichen mit den beiden folgenden Rollen stehen ihm

[7]Es existiert auch noch eine vierte Rolle *Goalie*, dieser gibt seine Rolle jedoch niemals ab und wird deshalb nicht weiter betrachtet.

dafür eine große Zahl von Aktionen zur Verfügung. Das oben beschriebene Verhaltensnetzwerk sorgt dafür, die jeweils passende auszuwählen.

- *Support*: Besitzt ein Spieler diese Rolle, so hat er die Aufgabe, sich so zu positionieren, daß er den aktiven Spieler unterstützen kann. Ist dieser im Ballbesitz, so orientiert sich der unterstützende Spieler in Richtung gegenerisches Tor, um so z.B. ein Paßspiel zu ermöglichen. Bei einem gegnerischen Angriff bleibt er leicht versetzt hinter dem Aktiven und wird seinerseits aktiv, falls dieser den Gegner nicht aufhalten kann.
- *Strategic*: Die Libero-Rolle: Der *Stratege* hält sich immer nahe des eigenen Tores auf und sichert das Spiel nach hinten ab.

5.5.3 Dynamische Positionen

Jeder Rolle wird jeweils eine dynamische Position zugeordnet, die angibt, wo auf dem Spielfeld der Roboter seine Rolle am besten ausfüllen kann. Die Berechnung dieser Positionen erfolgt dabei auf der Basis der in Kapitel 3.3 beschriebenen erweiterten Auswertung des Weltmodells mit Potentialfeldern. Jede Rolle besitzt ein eigenes Positionspotentialfeld. Für jede Zelle eines solchen Feldes wird ihr Potentialwert bestimmt als gewichtete und normierte Summe verschiedener Einflußfaktoren. Jeder Einflußfaktor bildet ein Teilpotentialfeld. Als momentan beste Position wird der Mittelpunkt derjenigen Zelle mit dem kleinsten Potentialwert ausgewählt[8]
Um einerseits Rechenzeit zu sparen und andererseits bestimmte „absolute“ Randbedingungen zu berücksichtigen (beispielsweise sollen Supporter und Stratege niemals eine Position im Strafraum anfahren), wird in der Regel nur ein Teil des Spielfelds betrachtet. Es folgt eine Beschreibung der drei Potentialfelder f_{active}, $f_{support}$, $f_{strategic}$:

Potentialfeld $\mathbf{f_{active}}$: Dieses Potentialfeld ist nur dann relevant, wenn der aktive Spieler nicht in Ballbesitz ist und er mit Hilfe des Pfadplaners eine günstige Position in der Nähe des Balles anfahren möchte. Die günstigste Position (berücksichtigt man nur den Spieler und den Ball) wird dabei charakterisiert durch eine bestimmte Mindestdistanz zum Ball und einen optimalen Winkel zu diesem. Bei der Berechnung fließen folgende Faktoren ein:

- Die Position soll so gewählt werden, daß sich eine möglichst geringe Distanz zum Ball ergibt.
- Dieses Teilpotentialfeld gibt jeweils den optimalen Winkel zum Ball vor und bewertet Positionen um so schlechter, je mehr sie von diesem abweichen. Befindet sich der Ball nahe dem eigenen Tor, so ergibt sich der optimale Winkel aus der Verlängerung [Mitte der eigenen Torlinie → Ballposition], im Mittelfeld wird dagegen die Verlängerung [Ballposition →

[8] Dieser Ansatz orientiert sich am *SPAR*-Ansatz (*Strategic Positioning with Attraction and Repulsion*) in [Veloso *et al.*, 1998].

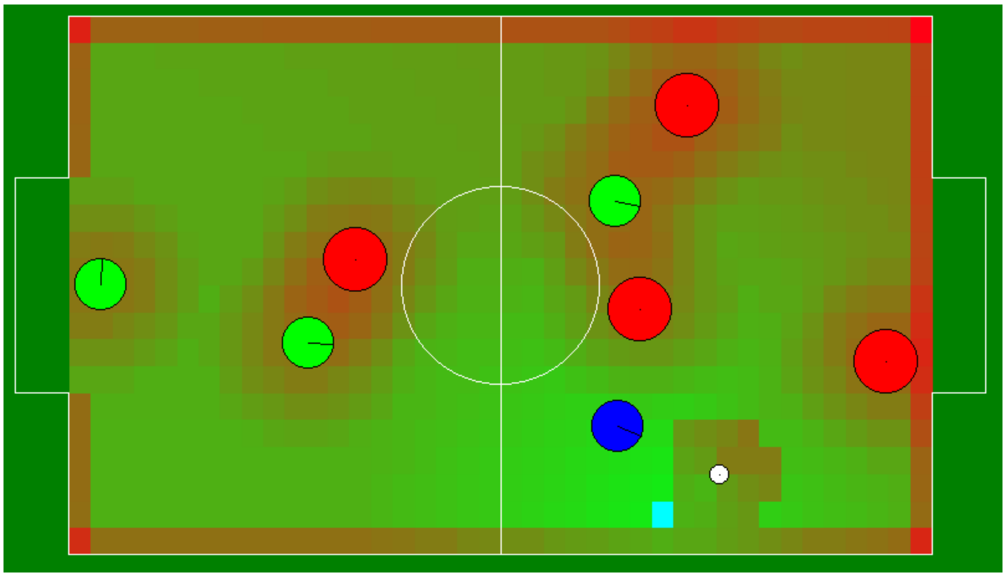

Abbildung 5.9: Das Potentialfeld f_{active}.

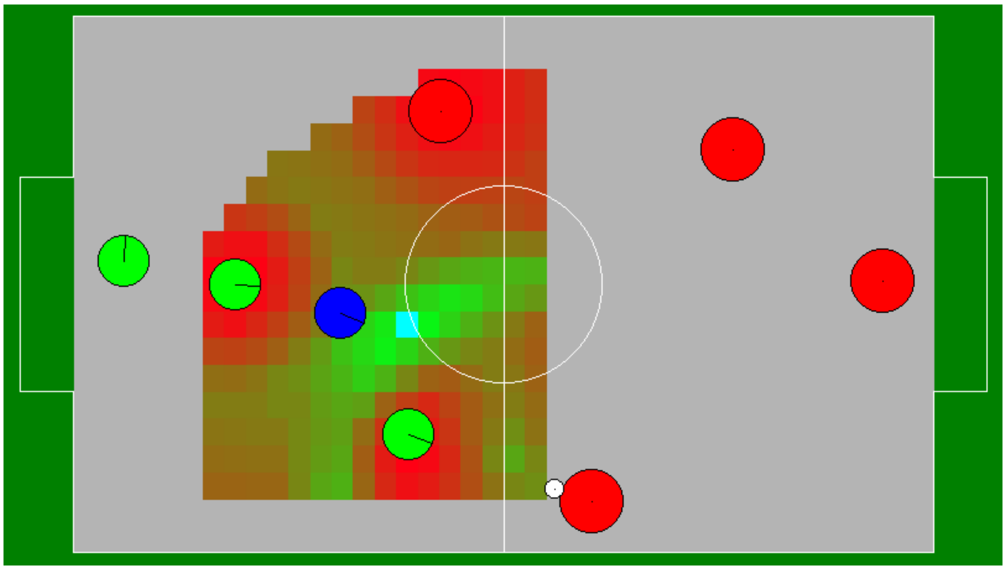

Abbildung 5.10: Das Potentialfeld $f_{support}$ (defensive Variante).

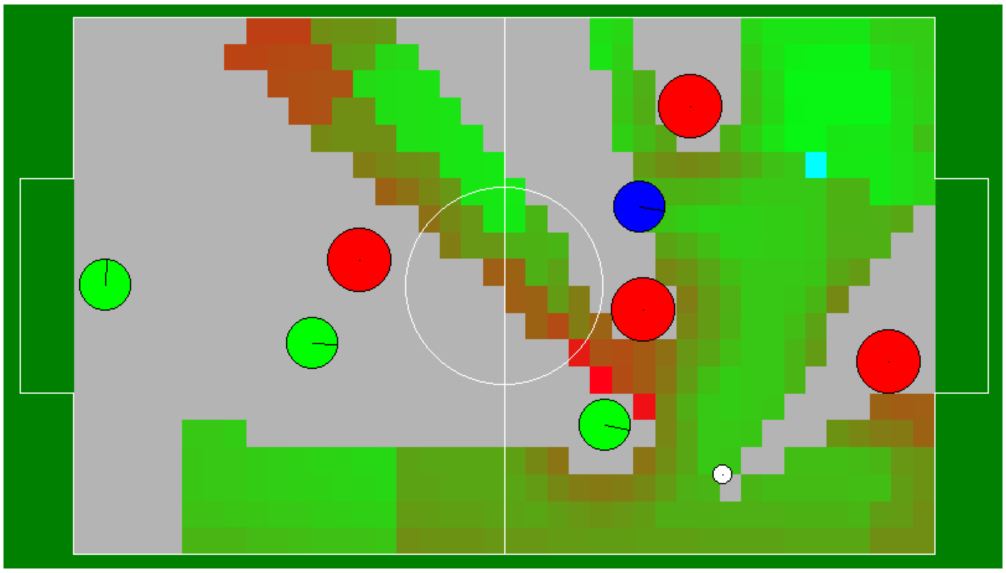

Abbildung 5.11: Das Potentialfeld $f_{support}$ (offensive Variante).

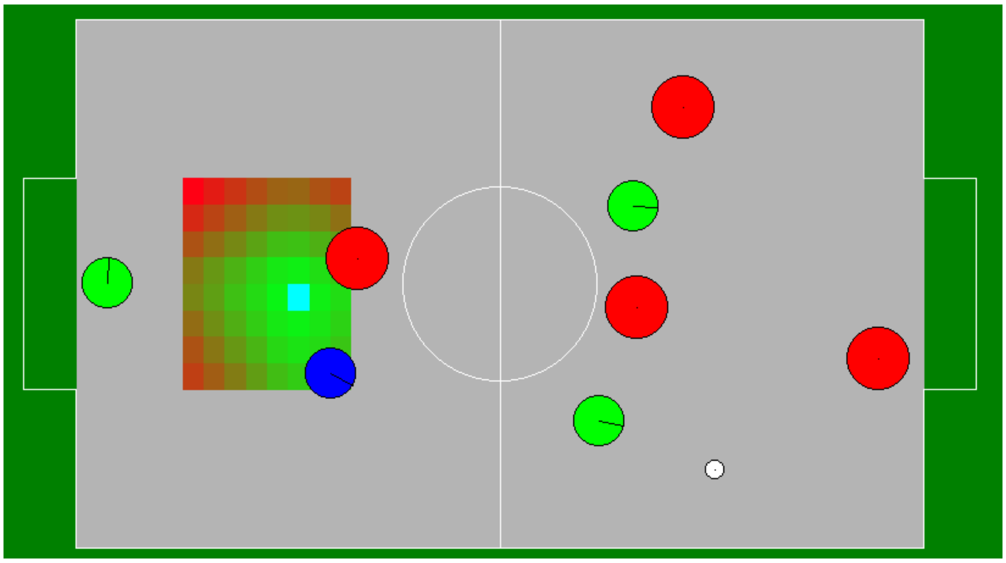

Abbildung 5.12: Das Potentialfeld $f_{strategic}$.

Mitte der gegnerischen Torlinie] als optimal angenommen. Und in der Nähe des gegnerischen Tors orientiert sich der optimale Winkel an der freiesten Stelle auf der gegnerischen Torlinie.

- Der Ball ist von einem abstoßenden Potential umgeben, damit der Roboter mit Planer zunächst nicht zu dicht an den Ball heranfährt und ihn beim Manövrieren unabsichtlich wegstößt.
- Die Spielfeldbegrenzungen haben eine *abstoßende* Wirkung, d.h. je näher sich eine Position an der Bande befindet, desto höhere Werte enthält dieses Teilpotentialfeld. Dies verhindert, daß der Roboter sich selbst manövrierunfähig macht, falls er zu dicht an der Bande entlangfährt.
- Ebenso hat der Einflußbereich von Gegnern und Mitspielern eine abstoßende Wirkung, um Zusammenstößen vorzubeugen. Die entsprechenden Werte können dabei direkt aus den Potentialfeldern $f_{covered}$ und f_{free} (siehe S. 19) abgelesen werden.
- Es werden solche Positionen bevorzugt, deren Winkel zum Ball wenig abweicht von dem der aktuellen Roboterposition zum Ball.

Bei der Bestimmung der *Active*-Position wird nur ein etwa 1,5 mal 1,5 Meter^2 großer Ausschnitt des Spielfelds um den Ball ausgewertet.
Abbildung 5.9 zeigt das Potentialfeld f_{active} für eine Spielsituation. Die eben genannte Einschränkung wurde für diese Darstellung aufgehoben. Die optimale *Active*-Position ist magenta-farben markiert. Ansonsten entspricht die Farbe rot hohen, Zellen grüner Farbe dagegen niedrigen Potentialwerten. Der aktuell betrachtete Spieler hat eine blaue Farbe, Mitspieler sind grün markiert, Gegenspieler rot.
Die Berechnung der momentan besten *Active*-Position läßt sich formal beschreiben durch[9]

$$f^t_{active}(x,y) = \frac{\sum_{i=1}^{7} w_i \cdot f_i^t(x,y)}{\sum_{i=1}^{7} w_i},$$

mit $w_i \in [0..1]$ und

$$f_1^t(x,y) = up\_straight(distance\_to\_ball^t(x,y), \\ min\_ball\_dist, min\_ball\_dist + ball\_dist\_range),$$

$$f_2^t(x,y) = up\_straight(|NormAngle(angle\_to\_ball^t(x,y) - opt\_angle)|, \\ angle\_tolerance, angle\_tol\_range),$$

$$f_3^t(x,y) = straight\_down(distance\_to\_ball^t(x,y), \\ min\_ball\_dist, min\_ball\_dist + ball\_safetydist\_range),$$

$$f_4^t(x,y) = straight\_down(distance\_to\_border^t(x,y),$$

[9]Zu den Funktionen *up_straight* und *straight_down* siehe S. 42. Die Funktion $k : N \to \mathbb{R}^{\nvDash}$ bildet den Index einer Zelle auf ihren Mittelpunkt ab.

$$min\_border\_dist, border\_safetydist\_range),$$

$$f_5^t(x,y) = (1 - f_{free}^t(x,y)),$$

$$f_6^t(x,y) = f_{covered}^t(x,y),$$

$$f_7^t(x,y) = up\_straight(|NormAngle(angle\_to\_ball^t(x,y) - angle\_to\_ball^t(robot^t.x, robot^t.y))|, 0.0, turn\_angle\_range)$$

Und somit:

$$\mathcal{C} = \{cell|\ \ |k(cell).x - ball^t.x| < 750.0 \wedge |k(cell).y - ball^t.y| < 750.0\},$$

$$cell_{min}^t = arg\min_{cell \in \mathcal{C}} f_{active}^t(k(cell)),$$

und

$$activepos^t = k(cell_{min}^t).$$

Potential $\mathbf{f_{support}}$: Wie schon angedeutet, gibt es für die *Support*-Position eine defensive und eine offensive Variante.
Die defensive Variante setzt sich wie folgt zusammen:

- Es wird ein optimaler Defensivwinkel zum Ball bestimmt, von dem möglichst wenig abgewichen werden soll. Je nachdem, ob sich der Ball gerade in der linken oder rechten Hälfte des Spielfelds befindet, soll der Spieler zur Mitte hin um 30 Grad versetzt hinter dem Ball stehen.

- Die Distanz des unterstützenden Spielers zum Ball soll nicht zu gering sein, damit er den aktiven Mitspieler nicht behindert, aber auch nicht zu groß. Daher wird eine beste Balldistanz vorgegeben und ein Teilpotential eingeführt, das Positionen benachteiligt, je mehr ihre Balldistanz von dieser abweicht.

- Die Einflußbereiche von Gegnern und Mitspielern haben eine abstoßende Wirkung.

- Die zurückzulegende Strecke zwischen jetziger Position des Roboters und optimaler Position soll minimiert werden.

- Positionen, die der Roboter ohne große Orientierungsänderungen anfahren kann, werden bevorzugt. Dieser Einflußfaktor, der sich in ähnlicher Form in allen Positionspotentialfeldern und in den weiter unten beschriebenen *Utility*-Funktionen wiederfindet, hängt mit der Beweglichkeit der Freiburger Roboter zusammen. Während andere Teams häufig schneller und wendiger sind oder sogar einen omnidirektionalen Antrieb besitzen, sind Positionierungsmanöver für die Freiburger Spieler häufig mit einigem Zeitaufwand verbunden — es sei denn, die momentane Orientierung stimmt schon in etwa mit der Richtung der Zielposition und der Zielorientierung überein.

Von der Berechnung ausgenommen werden hier solche Positionen, die weiter vom Ball entfernt sind als eine bestimmte maximale Distanz, außerdem Positionen innerhalb des Strafraums und solche, die sich sehr nahe an der Bande oder vor dem Ball in Richtung gegnerisches Tor befinden. Abbildung 5.10 stellt die defensive Variante von $\mathbf{f_{support}}$ dar.
Ein Beispiel für eine offensive Positionierung des unterstützenden Spielers zeigt Abbildung 5.11. Wiederum werden Positionen jenseits einer betimmten Balldistanz nicht betrachtet, genausowenig wie Positionen, von denen aus der direkte Weg zum Ball durch ein Hindernis versperrt wird. Außerdem kommen solche Positionen nicht in Frage, die einen direkten Torschuß des aktiven Spielers blockieren würden (hierzu wird das Potentialfeld $f_{goalway}$ ausgewertet).
Die offensive Variante von $f_{support}$ enthält folgende Teilpotentiale:

- Der Spieler soll eine taktisch möglichst gute Position einnehmen. Wie gut eine Position taktisch prinzipiell ist, wird dabei direkt im Potentialfeld $f_{desired}$ abgelesen.
- Wie bei der defensiven Variante wird auch hier eine beste Distanz zum Ball vorgegeben. Sie ist hier jedoch größer und orientiert sich an einer optimalen Paßdistanz.
- Die Einflußbereiche von Gegnern und Mitspielern haben eine abstoßende Wirkung.
- Positionen in der Nähe der momentanen Position des Spielers werden bevorzugt.
- Der Roboter soll seine Ausrichtung möglichst wenig ändern.

Die Entscheidung darüber, ob der Spieler mit der Rolle *Support* sich defensiv oder offensiv orientieren soll, hängt ab von der Position des Balls, davon, welche Mannschaft in Ballbesitz ist, und — falls nicht der Gegner den Ball hat — vom Abstand des nächsten Gegners zum Ball. Je weiter sich der Ball in der gegnerischen Hälfte befindet, desto eher wird der Supporter offensiv werden, auch wenn der aktive Mitspieler noch nicht in Ballbesitz ist, dieser aber näher zum Ball steht als ein Gegenspieler. In der Nähe des eigenen Tores bleibt der unterstützende Spieler stets defensiv; im Mittelfeld orientiert er sich nur dann nach vorne, falls die eigene Mannschaft den Ball hat.

Beim RoboCup-Turnier wurde die offensive Supportposition nicht benutzt. Zum einen war Paßspiel aufgrund eines fehlenden Annahmeverhaltens nur sehr eingeschränkt möglich, zum anderen mußte der CS Freiburg über weite Strecken des

Turniers die Taktik einer defensiven Kontermannschaft spielen. Daher wurde stets eine defensive *Support*-Position vorgegeben und die Berechnung so abgewandelt, daß der Spieler nur bis in die Nähe der Mittellinie aufrücken konnte. Zusätzlich existierte eine spezielle Variable, mit der es dem Supporter erlaubt wird, direkt eine der beiden festgelegten Paßpositionen (links und rechts vor dem gegnerischen Tor) anzufahren, falls sich der Ball in der Nähe des gegnerischen Tores befindet und der Gegner nicht in Ballbesitz ist.

Potentialfeld $\mathbf{f_{strategic}}$: Dieses Potentialfeld schließlich setzt sich aus folgenden Teilpotentialen zusammen:

- Es wird jeweils eine optimale, orthogonal zu den Seitenbegrenzungen verlaufende Linie für den Strategen bestimmt, deren Abstand minimiert werden soll. Die optimale Linie ergibt sich dabei aus der x-Koordinate des am nächsten zum eigenen Tor stehenden Gegenspielers, aus der x-Koordinate eines aktiven Mitspielers (der Stratege soll sich jeweils näher als diese zum eigenen Tor befinden) sowie aus der x-Koordinate des Balls (je weiter der Ball in der gegnerischen Hälfte ist, desto mehr darf der Stratege aufrücken).
- Der Abstand zur Mitte des Spielfelds (d.h. zur Mittellinie parallel zu den Seitenbanden) soll ebenfalls minimiert werden.
- Der Spieler möchte einen möglichst großen Abstand zum gerade aktiven Mitspieler halten, um diesen nicht zu behindern.
- Je geringer der Abstand der fraglichen Position von der momentanen Position des Roboters, desto geringer d.h. vorteilhafter sind die Werte dieses Teilpotentials.

Wie aus Abbildung 5.12 hervorgeht, ist die dynamische *Strategic*-Position auf einen relativ kleinen Bereich des Spielfelds beschränkt. Während des *RoboCup*-Turniers zeigte sich, daß es besser ist, den Strategen nicht zu sehr zur Seite hin „abdriften"zu lassen. Positionen, die außerhalb eines durch die Torpfosten markierten „Schlauchs" liegen, werden daher gar nicht erst betrachtet, ebensowenig wie solche jenseits der optimalen x-Koordinate (s.o.). Wie bei der *Support*-Position wird auch der Strafraum ausgeklammert.

5.5.4 Utility-Funktionen

Um nun zu bestimmen, welcher Agent welche Rolle einnimmt, berechnet jeder Spieler eine Einschätzung, wie gut er glaubt, eine bestimmte Rolle momentan ausführen zu können. Dazu stehen drei sogenannte *Utility-Funktionen* zur Verfügung, die jeweils auf der Basis der Daten des Weltmodells einen Wert aus [0..1] liefern. Die Utility-Werte ergeben sich dabei wiederum als normierte, gewichtete Summe verschiedener Merkmale:

Wichtigstes Merkmal bei der Berechnung der $\textbf{utility}_{\textbf{active}}$ ist die Distanz des Spielers zum Ball, die natürlich möglichst klein sein soll. Daneben fließt der Winkel der Spielerposition zur Ballposition mit ein, d.h. inwieweit sich der Spieler vom eigenen Tor aus gesehen auf der „richtigen" Seite des Balles steht. Dieses Merkmal hat jedoch weniger Gewicht, wenn sich der Ball nahe der Bande befindet. Schließlich werden die Spieler bevorzugt, die ihre Ausrichtung möglichst wenig ändern müssen, um sich zum Ball zu orientieren.
Die Berechnung der $\textbf{utility}_{\textbf{support}}$ und $\textbf{utility}_{\textbf{strategic}}$ berücksichtigt die nach dem oben beschriebenen Verfahren bestimmte, momentan beste *Support*- bzw. *Strategic*-Position. Bewertet werden die Distanz des Spielers zur Position und wieweit er seine Orientierung ändern müßte, um sie anzufahren.

5.5.5 Kooperative Rollenverteilung

Algorithmus 5.1 ist eine allgemeine Formulierung des Verfahrens zur Verteilung der Rollen (vgl. [Castelpietra *et al.*, 2000]). In unserem Fall gibt es drei Feldspieler und drei Rollen, d.h. $\mathcal{R}$ = {Otto, Jupp, Berti} und $\rho = \{Active, Strategic, Support\}$[10].

Algorithmus 5.1 *UpdateRoles*
Gegeben:
  Menge von Robotern $\mathcal{R} = \{R_1, \dots, R_n\}$
  Menge von Rollen $\rho = \{\rho_1, \dots, \rho_m\}$,
    mit ρ_i wichtiger als ρ_{i+1}.
  Persistenzwerte $persistence : \rho \to [0..1]$
  Default-Rollen $default\_roles : \mathcal{R} \to \rho$
Daten:
  $current\_role : \mathcal{R} \to \rho$,
  $desired\_role : \mathcal{R} \to \rho$,
  $utilities$: $\mathcal{R} \times \rho \to [0..1]$.
Eingabe: robot : $\mathcal{R}$
Ausgabe: *current_role*(robot), *desired_role*(robot) : ρ
Ablauf:
  for each $R_i \in \mathcal{R}, R_i \neq robot$ **do**
    $current\_role(R_i) = receive\_current(R_i)$
    $desired\_role(R_i) = receive\_desired(R_i)$
    for each $\rho_j \in \rho$ **do**
      $utilities(R_i, \rho_j) = receive\_utility(R_i, \rho_j)$
  for each $\rho_j \in \rho$ **do**
    $utilities(robot, \rho_j) = calculate\_utility(robot, \rho_j)$

  desired_role(robot) = **DetermineDesiredRole**(robot)
  if RoleAvailable(robot, *desired_role*(robot))

[10] Für eine eher defensiv ausgerichtete Mannschaft wie den *CS Freiburg* führte diese Rollenpriorisierung in der Regel zu besserem Stellungsspiel als $\rho = \{Active, Support, Strategic\}$.

current_role(robot) = *desired_role*(robot)
for each $\rho_j \in \rho$ **do**
send(utility(robot, ρ_j))
send(*current_role*(robot)
send(*desired_role*(robot)

Funktion: *RoleAvailable*(robot : $\mathcal{R}$, role : ρ) : $\{TRUE, FALSE\}$
for each $R_i \in \mathcal{R}, R_i \neq$ robot **do**
if *current_role*(R_i) == role && *desired_role*(R_i) == role
return FALSE
return TRUE

Funktion: *DetermineDesiredRole*(robot : $\mathcal{R}$) : ρ

$\mathcal{A} = \mathcal{R}$ /* not yet assigned robots */
for each $\rho_i \in \rho$ do
for each $R_j \in \mathcal{A}$ **do**
if *current_role*(R_j) == ρ_i
$utilities(R_j, \rho_i) = utilities(R_j, \rho_i) \cdot (1 + persistence(\rho_i))$
best_robot = $arg\max_{R_j \in \mathcal{A}}(utilities(R_j, \rho_i))$
if best_robot == robot
return ρ_i
$\mathcal{A} = \mathcal{A} \setminus$ best_robot
return *default_role*(robot)

Grundlage des Verfahrens ist, daß jeder Spieler allen anderen seine Utility-Werte mittteilt und auch, welche Rolle er gerade einnimmt und welche Rolle er gerne einnehmen möchte.
Anhand der eigenen Utility-Werte und der seiner Mitspieler bestimmt jeder Spieler in jedem Zyklus zunächst seine *gewünschte* Rolle (*desired role*), d.h. diejenige Rolle höchster Priorität, für die er die größte *Utility* besitzt. Dabei fließt ein sog. *Persistenzwert* der Rolle mit ein: Dieser gibt an, um wieviel der Utility-Wert eines Spielers für eine Rolle erhöht wird, falls er diese Rolle gerade innehat. Anders ausgedrückt: Je höher der Persistenzwert, desto schwieriger ist es für die Mitspieler, den Rolleninhaber abzulösen. Rollenpersistenz dient dazu, oszillierende Rollenzuweisungen zu vermeiden.
Als nächstes prüft jeder Agent, ob die gewünschte Rolle auch *verfügbar* (*available*) ist. Verfügbar ist eine Rolle dann, wenn kein Mitspieler existiert, der die Rolle gerade ausführt und diese auch beibehalten möchte. Nur falls seine gewünschte Rolle verfügbar ist, nimmt der Agent sie ein. Ansonsten behält er einfach die vorige Rolle bei. Diese Vorgehensweise hat zum Ziel, „chaotische“ Rollenwechsel zu verhindern, und Situationen, in denen mehrere Spieler die gleiche Rolle einnehmen.

5.5.6 Diskussion

Vergleicht man das verwendete Verfahren mit dem in Kapitel 2.3.2 vorgestellten *Contract Net Protokoll*, so lassen sich einige Ähnlichkeiten erkennen. Beim Multiagentensystem *CS Freiburg* gibt es stets die gleichen Aufgaben zu verteilen: „Wer geht zum Ball, wer unterstützt, wer spielt Libero?“ Natürlich möchte jeder der Spieler gerne zum Ball gehen und Tore schießen. Die Angebote der Agenten in Form der Utility-Werte entscheiden jedoch, wer welche Rolle übernimmt. Im Unterschied zum klassischen *Contract Net*-Protokoll gibt es jedoch keinen *Manager*, der jeweils die Aufgaben verteilt. Es wäre durchaus möglich gewesen, die Kompetenz des Rechners außerhalb des Spielfelds dahingehend zu erweitern, daß er die Angebote aller Spieler entgegennimmt und anschließend jedem seine Rolle zuweist. In Kapitel 4.2 wurde angesprochen, daß einige RoboCup-Teams den Ansatz zentraler Aufgabenverteilung wählen. Mit dem vorliegenden Verfahren soll jedoch die Autonomie der Agenten hervorgehoben werden. Jeder Spieler entscheidet für sich, welche Rolle er einnimmt, beachtet dabei aber die Angebote seiner Mitspieler und die gemeinsamen Regeln (vgl. auch Kapitel 4.2, *locker-room agreements*). Man kann so vielleicht von einer vereinfachten, „eigenverantwortlich kooperativen“ Variante des *Contract Net* Protokolls sprechen.

Das vorgestellte Konzept der neuen Mannschaftstaktik ermöglicht es den Spielern des *CS Freiburg*, deutlich genauer und flexibler auf unterschiedliche Spielsituationen zu reagieren. Das Konzept dynamischer Rollen kann bei Bedarf leicht um weitere Rollen erweitert werden. Ein weiterer Schritt könnte sein, die Menge der Rollenvorgaben dynamisch zu verändern und so auch verschiedene Formationen zu realisieren (vgl. Kapitel 4.2). Die modulare Darstellung der dynamischen Positionsberechnung als gewichtete Summe von Potentialfeldern führt zu einer guten Handhabung und ermöglicht die Verwirklichung einer weiten Bandbreite unterschiedlicher Stellungsspielvarianten.
Vorteilhaft erscheint auch, daß die verschiedenen Absprachen, Nachrichten und Regeln des Vorjahres nun in ein Gesamtkonzept integriert wurden. Zudem konnte im beschriebenen Verhaltensnetzwerk durch die Möglichkeit dynamischer Ziele die konzeptuelle Trennung von Rollenverteilung und Aktionsauswahl elegant abgebildet werden.
Allerdings soll auch der Nachteil der höheren Flexibilität und der gewachsenen Ausdrucksmöglichkeiten nicht verschwiegen werden: Eine ganze Anzahl neuer Parameter und Gewichte muß (zumindest im Moment noch) in mühevoller Kleinarbeit manuell optimiert werden.

5.6 Ergebnisse

Der Gewinn des Weltmeistertitels beim *RoboCup 2000* macht deutlich, daß die verschiedenen Neuentwicklungen, Umgestaltungen und Verbesserungen auch in diesem Jahr wieder zu einem robusten und sehr spielstarken Gesamtsystem geführt haben.

Den Grundstein für diesen Erfolg hat ohne Zweifel die von der Ausbildungswerkstatt der Firma SICK neu gestaltete Schußvorrichtung und die damit verbundene Überarbeitung unter anderem der Motoransteuerung gelegt. Ein neues Material an der Oberfläche der Schußvorrichtung führte außerdem zu einer stark verbesserten Ballführung. Auf dieser Grundlage konnten neue, sehr erfolgreiche Grundverhalten wie z.B. *Turn-Ball*, *Dribble-Ball* oder der „Bandenkick“ implementiert werden.
Aus der Sicht der Strategiekomponente ist festzustellen, daß die Umstellung auf den Algorithmus der Erweiterten Verhaltensnetzwerke erfolgreich war. Verglichen mit dem Spiel anderer Mannschaften zeichnet sich das Freiburger Spiel vor allem durch flüssige Bewegungsabläufe, hohe Präzision und den flexiblen Einsatz einer ganzen Anzahl von Grundverhalten aus. Einige sehr schöne Aktionssequenzen konnten während des Turniers beobachtet werden. Zu all dem leistete das in diesem Kapitel beschriebene Verhaltensnetzwerk einen Beitrag.
Bei der Mannschaftstaktik waren deutliche Verbesserungen durch das Konzept dynamischer Rollen und Positionen erkennbar. Nicht nur führte die Möglichkeit dynamischer Rollenwechsel zu einer insgesamt effizienteren Aufgabenteilung. Die dynamisch berechneten Positionen ermöglichten ein geradezu „quirliges“ Stellungsspiel, so daß sich Angriffsmannschaften wie *Sharif CE* (Iran) oder das von ihrer Roboterhardware eigentlich überlegene *Golem Team* (Italien) diesmal an der Freiburger Defensive „die Zähne ausbissen“.
Leider konnte das Paßspiel in diesem Jahr nicht rechtzeitig fertiggestellt werden. Tests ergaben, daß aufgrund der hohen Schußstärke der neuen Schußvorrichtung ein exaktes Zuspiel sowie eine erfolgreiche Annahme bzw. Weiterleitung mittels *Flipper-Shoot* schwierig ist.

Die Veränderungen der Strategiekomponente empirisch zu evaluieren (z.B. durch Spielreihen mit 2 gegen 2 Robotern), konnte im Rahmen dieser Arbeit nicht mehr durchgeführt werden. Ein Spiel der letztjährigen Strategiekomponente gegen die diesjährige scheitert an der Anzahl der zur Verfügung stehenden Roboter. Ein empirisches Maß, das bis zu einem gewissen Grad Aufschluß über die Spielstärke einer Mannschaft geben kann, ist die *Torrate* ($\frac{Tore}{min}$). Abbildung 5.13 zeigt zunächst die Verteilung der Torrate beim diesjährigen *RoboCup*. Freiburg befindet sich mit 0,215 Toren pro Minute in der Spitzengruppe (neben den *Osaka Trackies* mit 0,222 $\frac{Tore}{min}$, dem *Golem Team* mit 0,21 $\frac{Tore}{min}$, der Mannschaft des *RMIT Melbourne* mit 0,181 $\frac{Tore}{min}$ und *Sharif CE* mit 0,175 $\frac{Tore}{min}$). Man erkennt eine *bimodale Verteilung*: Einige Mannschaften scheinen alle anderen zu dominieren. Die Entwicklung der durchschnittlichen Torrate des Freiburger Teams verglichen mit der aller Teams über die letzten drei RoboCup-Wettbewerbe 1998, 1999 und 2000 zeigt Abbildung 5.14. Während sich die durchschnittlich erzielten Tore pro Spielminute aller Mannschaften 1998 und 1999 nur unwesentlich unterschieden (0,077 $\frac{Tore}{min}$ bzw. 0.073 $\frac{Tore}{min}$) ist von 1999 auf 2000 eine deutliche Steigerung zu verzeichnen (auf 0,114 $\frac{Tore}{min}$). Die Torrate des *CS-Freiburg* steigerte sich von 0,2 $\frac{Tore}{min}$ 1998 um 10 % auf 0,2 $\frac{Tore}{min}$ 1999 und ging in diesem Jahr wieder leicht zurück. Offensichtlich hatten die 1999 durchgeführten Verbesserungen einen größeren Einfluß auf die erzielten Tore pro Minute, was sich ja auch mit der Beobachtung von *Reetz* deckt, daß nun nicht mehr offensichtliche Torchancen aufgrund starrer Raumaufteilung vergeben wurden[1999]. Der Effekt der diesjährigen Verbesserungen zeigt sich dagegen eher in der Stabilisierung der durchschittlichen Torrate auf hohem Niveau, trotz der — im Vergleich zum Vorjahr — zumindest teilweise sehr viel stärkeren Gegner (die Gewinner der Plätze 2 und 3 des Turniers waren in der Freiburger Vorrundengruppe).

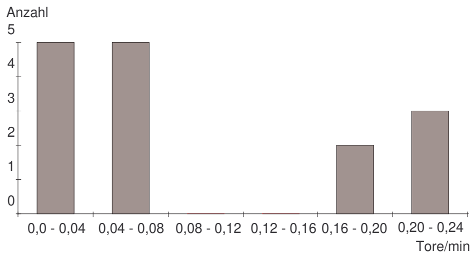

Abbildung 5.13: Verteilung der Torrate 2000.

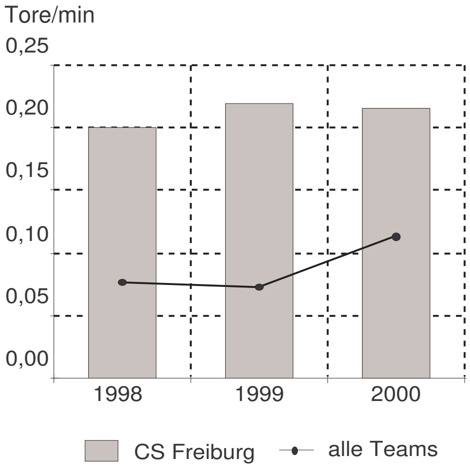

Abbildung 5.14: Entwicklung der Torrate.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Thema dieser Arbeit war die Strategiekomponente der Roboterfußballmannschaft *CS Freiburg*. Sie hat die Aufgabe, das Verhalten der Spieler zu steuern, indem jeweils aus einer Anzahl von Grundverhalten das der Spielsituation angemessene ausgewählt wird. Dieses übernimmt dann seinerseits die Robotersteuerung. Weiterhin ist sie für das Mannschaftsspiel der Roboter zuständig.
Nach einigen einführenden Bemerkungen zur*RoboCup*-Initiative wurde in Kapitel 2 ein kurzer Überblick über das Forschungsgebiet der Multiagentensysteme gegeben. Es wurden typische Eigenschaften von Agenten und Multiagentensystemen beschrieben und zwei zentrale Probleme herausgearbeitet: Das *Aktionsauswahlproblem* und das *Interaktionsproblem*. Anschließend wurden diverse Ansätze zur Lösung beider Probleme vorgestellt. Diese dienen dazu, die in dieser Arbeit behandelten Konzepte einzuordnen.
Kapitel 3 befaßte sich ganz allgemein mit der Hardware- und Softwarearchitektur des Multiagentensystems *CS Freiburg*.
Einige Ansätze anderer *RoboCup*-Mannschaften zu den beiden oben genannten Problemen wurden in Kapitel 4 vorgestellt.
Kapitel 5 beschrieb detailliert die im Rahmen dieser Arbeit implementierte Strategiekomponente. Dazu wurden zunächst einige Anforderungen an eine solche Komponente formuliert und anschließend — ausgehend von dem Konzept der letztjährigen Aktionsauswahl — die Verwendung der Erweiterten Verhaltensnetzwerke als Basis für das neue Aktionsauswahlmodul motiviert. Erweiterte Verhaltensnetzwerke wurden definiert und einige Eigenschaften diskutiert. Anschließend wurde auf die konkrete Implementierung eines Verhaltensnetzwerks für den diesjährigen *RoboCup*-Wettbewerb eingegangen. Wie sich während der Vorbereitung auf das Turnier gezeigt hat, können gegenwärtig die Möglichkeiten Erweiterter Verhaltensnetzwerke nur teilweise genutzt werden. Einige der sich u.a. durch die Struktur der bestehenden Grundverhalten ergebenden

Beschränkungen wurden analysiert. Der darauffolgende Abschnitt beschrieb das verbesserte kooperative Mannschaftsspiel. Die dynamischen Rollen sowie die Berechnung der dynamischen Positionen und das Verfahren zur Verteilung der Rollen wurden erläutert. Die Ergebnisse des diesjährigen *RoboCup*-Wettbewerbs, bei welchem der CS Freiburg den Weltmeistertitel erringen konnte, belegen, daß die im Rahmen dieser Arbeit vorgenommenen Veränderungen und Verbesserungen erfolgreich waren.

6.2 Ausblick

Damit der *CS Freiburg* auch zukünftig zu den Spitzenteams im Roboterfußball gehört, sollen abschließend einige Verbesserungsmöglichkeiten diskutiert werden, die sich aus den Erfahrungen mit dem System vor und während des RoboCup 2000 ergeben haben.

Profitieren würde die Strategiekomponente von einer Verbesserung der Ballpositionsschätzung. Darauf aufbauend könnten dann Grundverhalten und Strategien implementiert werden, die besser mit einem sich bewegenden Ball umgehen können als das aktuelle System.
Obwohl eine Revision des Schußmechanismus wohl in nächster Zeit nicht geplant ist, würde das Paßspiel davon profitieren, wenn dieser präziser und idealerweise mit regelbarer Schußstärke schießen würde.
Wie in Kapitel 5.4.3 gezeigt wurde, kann die Einbeziehung des Spielstands in die Verhaltensauswahl zu interessanten Effekten führen. Auch das Stellungsspiel könnte so erweitert werden, daß es je nach aktuellem Spielstand eher defensiv oder offensiv ist.
Allgemein bleibt es weiteren Arbeiten vorbehalten, die Möglichkeiten der Erweiterten Verhaltensnetzwerke stärker auszunutzen. Dazu gehört u.a. Verhaltenssequenzen noch stärker explizit im Netzwerk abzubilden, echt kontinuierliche Propositionen zu verwenden und Grundverhalten elementarer zu strukturieren. Im Verlauf dieser Arbeit wurde klar, daß die eher rigide, sich am Konzept der Aktionsmodule orientierende Struktur des vorgestellten Verhaltensnetzwerks das Mittel der Wahl darstellt, falls innerhalb kurzer Zeit ein funktionsfähiges System entstehen soll, dessen Komplexität durch die Vielzahl der Komponenten ohnehin groß genug ist. Besteht jedoch einmal mehr Raum für Experimente, so wäre es spannend, praktisch zu überprüfen, inwieweit tatsächlich Verhalten aus Verhaltensnetzwerken „emergieren“ kann.

Die in diesem Jahr eingeführten Neuerungen, nicht zuletzt die Positionspotentialfelder, führten zu mehr Flexibilität und Ausdruckskraft, hatten aber auch zur Folge, daß eine ganze Reihe neuer Parameter zu optimieren waren. Parameteroptimierung ist bei einem so komplexen System wie einem künstlichen Fußballspieler, der darüber hinaus noch real und nicht nur virtuell existiert, eine extrem zeitaufwendige und auch nervenaufreibende Aufgabe. Dies durch Lernverfahren zumindest teilweise zu vereinfachen, würde einen ganz erheblichen Fortschritt bedeuten.

Was das Stellungsspiel angeht, so agieren Feldspieler und Torwart derzeit noch relativ unabhängig voneinander. Eine Möglichkeit, das Stellungsspiel stärker aufeinander abzustimmen, wäre, eine besonders defensive Formation zu erstellen, und diese dynamisch immer dann einzusetzen, wenn der Torwart z.B. keine Nachrichten mehr sendet oder sich nicht bewegt.

Anhang A

Symbolerklärung

Die in den Verhaltensnetzwerk-Grafiken verwendeten Symbole haben folgende Bedeutung:

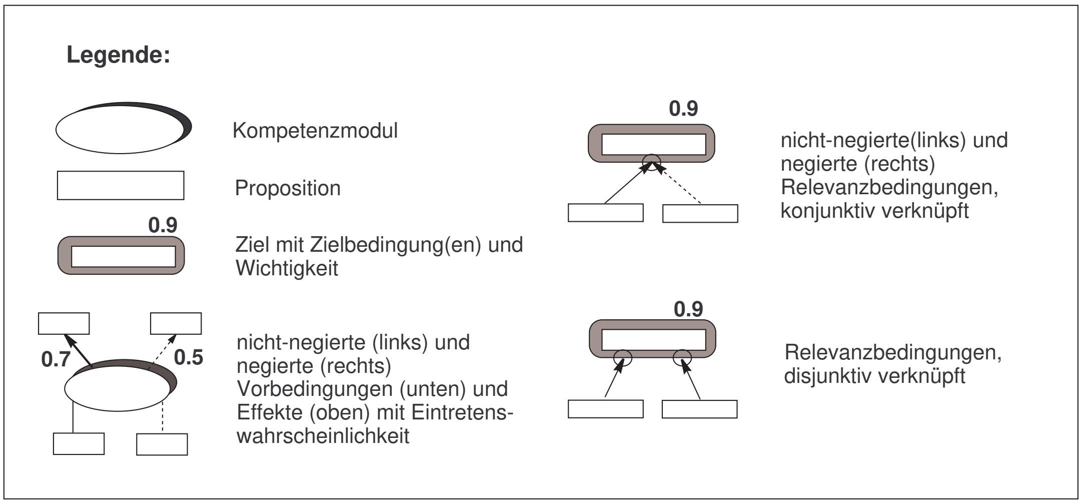

Abbildung A.1: Symbolerklärung zu den Verhaltensnetzwerk-Grafiken

Anhang B

Ergebnisse bei offiziellen Spielen

In diesem Kapitel werden die Ergebnisse der offiziellen Spiele des CS-Freiburg in den Jahren 1998, 1999 und 2000 in tabellarischer Form dargestellt.

Resultate 1998

Weltmeisterschaft RoboCup'98, Paris, 2.7-9.7.98:

CSF - NAIST	=	1 : 1 (nach Elfmeterschießen)
CSF - USC	=	1 : 0
CSF - Real Magicol	=	3 : 0
CSF - Yale	=	2 : 0
CSF - Osaka	=	3 : 0
CSF - Tübingen	=	2 : 0

CS-Freiburg ist Weltmeister 98.

Deutsche Meisterschaft VisionCup'98, Stuttgart, 30.9-1.10.98:

CSF - Ulm	=	4 : 0
CSF - Tübingen	=	4 : 1 (nach Elfmeterschießen)
CSF - München	=	2 : 0

CS-Freiburg ist Deutscher Meister 98.

Resultate 1999

Weltmeisterschaft RoboCup'99, Stockholm, 28.7-4.8.99:

CSF - ISocRob	=	5 : 0
CSF - GMD	=	6 : 0
CSF - KIRC	=	6 : 0
CSF - RMIT	=	6 : 0
CSF - Wisley	=	4 : 0
CSF - NAIST	=	4 : 1
CSF - ART	=	$0 : 0^+$
CSF - Wisley	=	2 : 1

CS-Freiburg gewinnt Bronze 99.

Deutsche Meisterschaft VisionCup'99, Stuttgart, 12-13.10.99:

CSF - GMD	=	6 : 0
CSF - Tübingen	=	4 : 0
CSF - München	=	$2^+ : 2$
CSF - Stuttgart	=	4 : 1

CS-Freiburg ist Deutscher Meister 99.

Die mit „+" gekennzeichnete Mannschaften haben das betreffende Spiel nach einer sog. technical challenge gewonnen. Dabei wird die Zeit gemessen, die ein Roboter benötigt, um mit einem im Spielfeld plazierten Ball ein Tor zu schießen.

Resultate 2000

Weltmeisterschaft RoboCup 2000, Melbourne, 28.8 - 4.9.2000:

CSF - NP Lions	=	8 : 0
CSF - CoPS Stuttgart	=	3 : 1
CSF - RMIT	=	3 : 0
CSF - KIRC	=	4 : 0
CSF - NAIST 2000	=	9 : 1
CSF - Sharif CE	=	5 : 1
CSF - Golem	=	1 : 2
CSF - CMU	=	5 : 0
CSF - Sharif CE	=	2 : 0
CSF - Golem	=	6 : 4 (nach Elfmeterschießen)
		(3 : 3 nach regulärer Spielzeit)

CS-Freiburg ist Weltmeister 2000.

Anhang C

Verhaltensnetzwerke

Die Spezifikation eines Verhaltensnetzwerks erfolgt in Form von Regeln: Ziele beginnen mit dem Schlüsselwort *Goal* und enden mit *endgoal*. Dazwischen wird die Relevanzbedingung als Bedingungsteil und die Zielbedingung mit der Wichtigkeit des Ziels als Konklusionsteil beschrieben. Entsprechend enthält eine Regel zur Beschreibung eines Kompetenzmoduls im Bedingungsteil die Vorbedingung, im Konklusionsteil das auszuführende Verhalten und nach dem Schlüsselwort *effect* dann die Effekte mit Erwartungswahrscheinlichkeit des Eintretens. Den Abschluß bildet das Schlüsselwort *endif*.

C.1 Das Netzwerk für *RoboCup 2000*

```
Params
gamma 0.5
delta 0.0
beta 0.1
theta 0.5
dtheta 0.1
endparams

Goal
if role_active
then soccergoal 1.0
endgoal

Goal
if not role_active
then cooperate 1.0
endgoal

If have_ball
```

```
and good_shoot_dist
and goal_free
and good_shoot_angle
and shoot_front_clear
and not shootgoal_exclusive_lastaction
then ShootGoal
effect soccergoal 0.8
endif

If have_ball
and front_clear
and goalarea_free
and good_play_angle
and good_moveshootfeint_dist
and not moveshootfeint_exclusive_lastaction
then MoveShootFeint
effect soccergoal 0.7
endif

If have_ball
and dribble_front_clear
and good_dribble_angle
and not dribbleball_exclusive_lastaction
then DribbleBall
effect soccergoal 0.6
endif

If have_ball
and turnball_angles_clear
and not turnball_exclusive_lastaction
then TurnBall
effect soccergoal 0.2
endif

If have_ball
and enemy_has_ball
and ball_near_own_goal
then WaitAndBlock
effect soccergoal 0.2
endif

If have_ball
and not turnawayball_exclusive_lastaction
then TurnAwayBall
effect soccergoal 0.1
endif
```

```
If not have_ball
and bumpshoot_ballway_free
and good_bumpoff_angle
and ball_in_opp_penalty_area
and good_bumpoff_ball_dist
then BumpShootOff
effect soccergoal 0.9
endif

If not have_ball
and ball_in_own_half
and good_bumpdef_angle
and enemy_closeto_ball
and good_bumpdef_ball_dist
and bumpshoot_ballway_free
and not ball_closeto_border
and allow_fast_defense
then BumpShootDef
effect soccergoal 0.9
endif


If not have_ball
and getball_ballway_free
and good_getball_angle
and good_getball_dist
then GetBall
effect have_ball 0.9
endif

If not have_ball
and getball_ball_close
then GetBall
effect have_ball 0.9
endif

If not have_ball
and ball_present
and not getball_ball_close
then GotoBall
effect good_getball_dist 0.9
and good_getball_angle 0.9
and getball_ballway_free 0.9
endif

If role_support
and not in_good_support_position
```

```
then GotoSupportPosition
effect cooperate 0.5
endif

If role_strategic
and not in_good_strategic_position
then GotoStrategicPosition
effect cooperate 0.5
endif

If ball_present
then ObserveBall
effect cooperate 0.2
endif

If not ball_present
then SearchBall
effect cooperate 0.6
and soccergoal 0.5
endif

If have_ball
and teammate_waitingfor_flipper
and good_flipperpass_position
and good_flipperpass_velocity
and not flipperpass_exclusive_lastaction
and not goal_free
then FlipperPass
effect soccergoal 0.85
endif

If role_allows_flippershoot
and in_good_flipper_position
and receive_pass_possible
then FlipperShoot
effect soccergoal 1.0
and cooperate 0.8
endif
```

C.2 Das Beispielnetzwerk aus Kapitel 5.4.3

```
Params
gamma 1.0
delta 0.0
```

```
beta 0.0
theta 0.1
dtheta 0.1
endparams

Goal
if not own_team_leading
then soccergoal 1.0
endgoal

Goal
if own_team_leading
then play_nice 0.8
endgoal

If have_ball
and good_shoot_dist
and goal_free
and good_shoot_angle
and shoot_front_clear
then ShootGoal
effect soccergoal 0.9
and play_nice 0.5
endif

If have_ball
and teammate_waitingfor_flipper
and good_flipperpass_position
then FlipperPass
effect soccergoal 0.4
and play_nice 0.9
endif
```

Abbildungsverzeichnis

Literaturverzeichnis

[Adorni *et al.*, 2000] G. Adorni, S. Cagnoni, M. Mordonini, und M. Piaggio. *Team/goal-keeper coordination in the RoboCup mid-size league.* In *The Fourth International Workshop on RoboCup*, Seiten 131–136, 2000.

[Agre und Chapman, 1990] P. Agre und D. Chapman. *What Are Plans for? Robotics and Autonomous Systems*, 6:17–34, 1990.

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

[Bylander, 1992] T. Bylander. *Complexity results for serial decomposability.* In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, Seiten 729–724, 1992.

[Castelpietra *et al.*, 2000] C. Castelpietra, L. Iochhi, D. Nardi, M. Piaggio, A. Scalzo, und A. Sgorbissa. *Communication and Coordination among heterogeneous Mid-size players: ART99.* In *The Fourth International Workshop on RoboCup*, Seiten 149–158, 2000.

[Chavez und Maes, 1996] A. Chavez und P. Maes. *Kasbah: An agent marketplace for buying and selling goods.* In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96)*, Seiten 75 – 90, 1996.

[Davis und Smith, 1983] R. Davis und R.G. Smith. *Negotiation as a metaphor for distributed problem solving. Artificial Intelligence*, 20:63–100, 1983.

[Dietl, 1999] Markus Dietl. *Globale Sensorintegration für RoboCup'99.* Studienarbeit, Universität Freiburg, 1999.

[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. Morgan Kaufmann, 1999.

[Dorer, 1999b] K. Dorer. *Motivation, Handlungskontrolle und Zielmanagment in autonomen Agenten.* Doktorarbeit, Universität Freiburg, 1999.

[Drücker *et al.*, 2000] C. Drücker, S. Hübner, E. Schmidt, Ubbo Wisser, und H.-G. Visser. *Virtual Werder: Using the Online-Coach to Change Team Formations.* In *The Fourth International Workshop on RoboCup*, Seiten 217–222, 2000.

[Ferguson, 1995] I.A. Ferguson. *Integrated control and coordinated behavior: A case for agent models.* In M. Wooldridge und N.R. Jennings, Herausgeber, *Intelligent Agents: Theories, Architectures and Languages (LNAI Volume 890)*, Seiten 203–218. Springer, Berlin, 1995.

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

[Flores-Mendez, 1999] R. Flores-Mendez. *Towards a Standardization of Multi-Agent System Frameworks.* http://www.acm.org/crossroads/xrds5-4/multiagent.html, ACM Crossroads, 1999.

[Franklin und Grasser, 1996] S. Franklin und A. Grasser. *Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents.* In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, 1996.

[Georgeff und Lansky, 1987] M.P. Georgeff und A.L. Lansky. *Reactive reasoning and planning.* In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seiten 677–682, 1987.

[Grand und Cliff, 1998] S. Grand und D. Cliff. *Creatures: Entertainment software agents with artificial life. Int. Journal of Autonomous Agents and Multi-Agent Systems*, 1(2), 1998.

[Gutmann *et al.*, 1999] J.-S. Gutmann, T. Weigel, und B. Nebel. *Fast, Accurate and Robust Self-Localization in the RoboCup Environment.* In *IJCAI-99: The Third International Workshop on RoboCup*, Seiten 109–114, 1999.

[Gutmann *et al.*, 2000] J.-S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, und T. Weigel. *The CS Freiburg Team: Playing Robotic Soccer Based on an Explicit World Model. The AI Magazine*, 2000.

[Herrman, 1998] Immanuel Herrman. *Roboter-Schußvorrichtung.* Technischer Bericht, Universität Freiburg, 1998.

[Huhns und Stephens, 1998] M.N. Huhns und L.M. Stephens. *Multiagent Systems and Societies of Agents.* In G. Weiß, Herausgeber, *Multi-Agent Systems.* The MIT Press: Cambridge, MA, 1998.

[Jäger und Christaller, 1997] H. Jäger und Th. Christaller. *Dual Dynamics: Designing behavior systems for autonomous robots.* In S. Fujimura und M. Sugisaka, Herausgeber, *Proceedings of the International Symposium on Artificial Life*, Seiten 76–79, 1997.

[Jennings *et al.*, 1998] N.R. Jennings, K. Sycara, und M. Wooldrige. *A Roadmap of Agent Research and Development. Int. Journal of Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[Kitano *et al.*, 1997] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, und E. Osawa. *RoboCup: A Challenge Problem for AI and Robotics.* In *RoboCup-97: Robot Soccer World Cup I*, Seiten 1–19. Springer-Verlag, 1997.

[Konolige *et al.*, 1997] K. Konolige, K. Myers, E. Ruspini, und A. Saffiotti. *The Saphira Architecture: A Design for Autonomy. Journal of Experimental and Theoretical Artificial Intelligence*, 9:215–235, 1997.

[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. Morgan Kaufmann, 1989.

[Maes, 1990] P. Maes. *Situated Agents Can Have Goals.* In P. Maes, Herausgeber, *Designing Atonomous Agents: Theory and Practice from Biology to Engineering and Back*, Seiten 49–70. MIT Press/Bradford Books, 1990.

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

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

[Müller, 1997] J.P. Müller. *The Design of Intelligent Agents (LNAI Volume 1177).* Springer, Berlin, 1997.

[Parunak, 1987] H. Van Dyke Parunak. *Manufacturing experience with the contract net.* In M .Huhns, Herausgeber, *Distributed Artificial Intelligence*, Seiten 285 – 310. Pitman Publishing: London, 1987.

[Rao und Wooldrige, 1999] A. Rao und M. Wooldrige. *Foundations of Rational Agency.* In A. Rao und M. Wooldrige, Herausgeber, *Foundations of Rational Agency*, Seiten 1–10. Kluwer Academic Publishers, 1999.

[Reetz, 1999] Christian Reetz. *Aktionsauswahl in dynamischen Umgebungen am Beispiel Roboterfußball.* Diplomarbeit, Universität Freiburg, 1999.

[Rosenschein und Kaelbling, 1990] S. Rosenschein und L. Kaelbling. *Action and Planing in Embedded Agents.* In P. Maes, Herausgeber, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, Seiten 35–48. MIT Press/Bradford Books, 1990.

[Russell und Norvig, 1995] S. Russell und P. Norvig. *Artificial Intelligence – A Modern Approach.* Prentice Hall, 1995.

[Scerri und Ydren, 1999] P. Scerri und J. Ydren. *End User Specification of RoboCup Teams.* In M. Veloso, E. Pagello, und H. Kitano, Herausgeber, *RoboCup-99: Robot Soccer World Cup III*, Seiten 450–459. Springer, 1999.

[Stone und Veloso, Juni 1999] P. Stone und M. Veloso. *Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence*, 2(110):241–273, Juni 1999.

[Sycara *et al.*, 1998] K. Sycara, K. Decker, und D. Zeng. *Intelligent Agents in Portfolio Management.* In N. Jennings und M. Woolridge, Herausgeber, *Agent Technology: Foundations, Applications, and Markets*, Seiten 267–283. Springer, 1998.

[Thiel, 1999] Maximilian Thiel. *Roboter-Fußball: Zuverlässige Ballerkennung und Positionsschätzung.* Diplomarbeit, Universität Freiburg, 1999.

[Topor, 1999] Augustinus Topor. *Roboterfußball: Pfadplanung in dynamischer Umgebung.* Diplomarbeit, Universität Freiburg, 1999.

[Veloso *et al.*, 1998] M. Veloso, M. Bowling, S. Achim, K. Han, und P. Stone. *The CMUnited-98 champion small robot team.* In *RoboCup-98: Robot Soccer World Cup II.* Springer-Verlag, 1998.

[Weigel, 1998] Thilo Weigel. *Roboter-Fußball: Selbstlokalisierung, Weltmodellierung, Pfadplanung und verhaltensbasierte Kontrolle.* Diplomarbeit, Universität Freiburg, 1998.

[Wooldridge, 1998] M. Wooldridge. *Intelligent Agents.* In G. Weiß, Herausgeber, *Multi-Agent Systems.* The MIT Press: Cambridge, MA, 1998.

[Zeng und Sycara, 1987] D. Zeng und K. Sycara. *Benefits of learning in negotiation.* In *Proceedings of the AAAI-97*, 1987.

[Zeng und Sycara, 1998] D. Zeng und K. Sycara. *Baysian learning in negotiation. International Journal of Human-Computer Studies*, 48, 1998.

Erklärung

Name: Klaus Müller Matr.Nr.: 9303624

Ich erkläre hiermit, daß ich diese Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Freiburg, den 30. Oktober 2000