technische universität
dortmund

Zwischenbericht

# Projektgruppe 529: Spielcharaktere - Modellierung menschenähnlicher Gegenspieler in Strategiespielen mit Techniken der CI

Niels Ackermann, Alireza Gholaman,
Marc Gorzala, Matthias Grochowski,
Thomas Harweg, Markus Kemmerling,
Daniel Spierling, Sebastian Uellenbeck,
Wolfgang Walz
8. August 2008

INTERNE BERICHTE

Betreuer:
Nicola Beume
Boris Naujoks
Mike Preuß

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.uni-dortmund.de

# Inhaltsverzeichnis

# 1 Einleitung

Die Projektgruppe 529 (PG529) befasste sich mit den Methoden der Computational Intelligence (CI) und den Einsatzmöglichkeiten in Computerspielen. Unter den verschiedenen Methoden der CI wurden innerhalb der PG529 vor allem Künstliche Neuronale Netze und Evolutionäre Algorithmen untersucht.

In vielen Computerspielen sind die NPCs (engl. Non-Player Character (NPC)) darauf ausgelegt eine bestimmte Zielfunktion zu maximieren bzw. zu minimieren, daher wirken sie oftmals sehr statisch und unmenschlich. Dies kann oft zu unerwünschten Effekten beim menschlichen Spieler führen, so dass zum Beispiel der Spielspass schnell verloren geht, da die Aktionen vom NPC leicht vorherzusagen sind und sich hiermit ein schneller Spielerfolg einstellen kann.

Die minimalen Ziele sah die Projektgruppe darin einen menschenähnlichen Computerspieler mit Hilfe der oben genannten CI-Techniken zu konstruieren, der nicht nach vorher festgelegten deterministischen Regeln seine Entscheidung trifft, sondern durch Lernverfahren sein Verhalten entwickelt. Dabei sollte vor allem untersucht werden, ob die Verwendung von CI-Methoden sich im Vergleich zu herkömmlichen Methoden der Künstlichen Intelligenz (KI) auszahlt, um den Minimalzielen gerecht zu werden.

Bevor die CI-Methoden an die Aufgabe angepasst und implementiert werden konnten, musste zunächst eine Wahl für ein bestimmtes Computerspiel getroffen werden. Die Computerspiele Diplomacy, Junta und Poker standen zur Auswahl. Für die Evaluierung spielten unter anderem die Benutzerfreundlichkeit, der Spielspass und die Kommunikationsmöglichkeiten der einzelnen Computerspiele eine wesentliche Rolle.

Nach der Wahl eines geeigneten Computerspiels folgte eine Planungsphase. In dieser Phase wurde das Projekt geplant und strukturiert. Anschließend folgte eine Implementierungsphase. Hier wurden die in der Planungsphase erarbeiteten Konzepte umgesetzt. Es zeigte sich, dass einige Fragestellungen offen geblieben waren, was zu erheblichen Zeitverzögerungen führte.

Kapitel 2 dieses Berichts enthält eine kleine Zusammenfassung der im Anhang ausführlich vorgestellten Seminararbeiten. Dies soll den Einstieg und den Umgang mit unserer Arbeit erleichtern.

Kapitel 3 beinhaltet verschiedene Implementierungsdetails und Probleme dieses Abschnitts. Weiter gibt Kapitel 3 einen Überblick über das erste Projekt und beschreibt zusätzlich unterschiedlichste Aspekte, von der Wahl von Poker als erstes Spiel, über die Organisation der Teilnehmer bis hin zur Wahl von C++ als Programmiersprache.

Nach Fertigstellung unserer Komponente wurde unser Spiel in einer Testphase von Probanden getestet. Am Ende von Kapitel 3 finden sich die Resultate dieses Tests.

Die Reflexionen, die während dieses einsemestrigen Projektes gesammelt wurden, sind im Kapitel 4 zusammengefasst.
Dieser Bericht schliesst mit einer Zusammenfassung (Kapitel 5) und einem Ausblick (Kapitel 6) in die Arbeit des zweiten Projekt-Semesters mit einem neuem Spiel dem sich die Projektgruppe widmet.

Im Anhang dieses Berichts befinden sich die Seminararbeiten und eine detaillierte übersicht der CI-Techniken.

# 2 Grundlagen

## 2.1 Evolutionäre Algorithmen

Evolutionäre Algorithmen(EA) stellen zufallsbasierte Optimierungsmechanismen dar, welche die biologische Evolution als Vorbild haben. Der Grundgedanke dabei ist, dass sich aus gefundenen Lösungen für das Optimierungsproblem ähnliche und hoffentlich auch bessere Lösungen ableiten lassen. In der Botanik verwendet man, um beispielsweise dem Ziel der Züchtung schnellerwachsender und mehr witterungsresistenter Getreidesorten näher zu kommen, zunächst Ableger einer Sorte, welche die gewünschten Eigenschaften noch nicht ausreichend besitzen. Ähnlich ist das Vorgehen bei EA.

Der Aufbau eines EA ist relativ simpel: Aus einer Anzahl von Startlösungen (im Fachjargon wird auch von einer *Population* gesprochen), die entweder zufällig oder auch gezielt mit anderen Heuristiken oder Approximationen bestimmt wurden, werden diejenigen ausgewählt, die sich als Vorbild für weitere gute Lösungen eignen. In Anlehnung an die Biologie, bezeichnet man bei EA eine Lösung auch als *Individuum*.

Für ein Optimierungsproblem muss eine Lösung bzw. ein *Individuum* bewertbar sein. Dies geschieht über eine Bewertungsfunktion, welche man auch als *Ziel-* oder *Fitnessfunktion* bezeichnet. Sie „vergibt“ an ein *Individuum* „Punkte“ in Abhängigkeit davon wie „gut“ dieses ist.
Mit der daraus berechneten *Fitness* lassen sich „gute“ Vorbilder, welche man auch als *Eltern* bezeichnet, auswählen, um nach deren Vorbild neue Individuen, auch *Kinder* genannt, zu erzeugen. Welche Individuen dafür geeignet sind, ist vor Erhalt der Ergebnisse nicht vorhersagbar. Man kann sie zufällig auswählen, wenn man überzeugt ist, dass man grundsätzlich aus einem „guten“ Pool von Individuen auswählt, oder sie nach deteministischen Regeln, wie „wähle die besten Individuen als neue Eltern aus“, bestimmt.

Für die Erzeugung der *Kinder* gibt es zwei grundlegende Konzepte:

- *Mutation*
- *Rekombination*

Bei der *Mutation* wird ein Kind, welches zuerst eine exakte Kopie seines Elters darstellt, durch Zufallseinfluss leicht verändert, so dass es sich nicht allzu sehr von dem Elter unterscheidet, jedoch auch nicht mehr identisch zu ihm ist.

Die *Rekombination* nimmt dagegen zwei (oder mehr) Elternindividuen und bildet aus zufällig ausgesuchten Teilen der Eltern ein neues Kindindividuum. Anschließend wird in der Regel noch das durch Rekombination gebildete Kindindividuum mutiert. Bei beiden Verfahren hat man die Hoffnung, dass gute Eigenschaften der Eltern erhalten bleiben oder sogar noch verstärken.

In manchen Fällen erweist sich das Bewerten eines *Individuum*s als sehr aufwendig, weshalb die Evaluierung aller neu erstellten Individuen in einem gesonderten Schritt durchgeführt wird.

Da ein Computer nur begrenzte Speicherkapazitäten hat und somit nicht unbegrenzt Daten erzeugen und verwalten kann, muss die Größe der *Population* beschränkt sein. Somit müssen einzelne bisher berechnete Individuen gelöscht werden. Dies sollten diejenigen Individuen sein, die wahrscheinlich nicht mehr zur Verbesserung der *Population* beitragen können.

Diese Frage lässt sich differenzierter betrachten, nämlich: in welcher Menge von Individuen werden die „überlebenden“ Individuen gesucht und wie werden sie ausgewählt.

Für die Frage nach der Menge der Individuen gibt es mehrere Alternativen. Beispielsweise sucht man aus der gesamten *Population* nach einer vorgegebenen Definition „gute“ Individuen aus. Dabei werden Individuen verschiedener Generationen gleich bewertet und haben (abgesehen von ihrem Fitness-Wert) die gleichen Chancen ausgewählt zu werden. Dieser Selektionstyp wird auch *Plus-Selektion* genannt.

Steht man auf dem Standpunkt, dass nur neue Individuen erhaltenswert sind, bietet sich die *Komma-Selektion* an. Dabei werden die Überlebenden nur unter den Kindern gesucht. Die Nebenbedingung, dass die Anzahl der Kinderindividuen mindestens so groß sein muss wie die Anzahl der Elternindividuen, ist daher ersichtlich, dass andernfalls nicht genügend Individuen zum „Auffüllen“ der Menge der Eltern vorhanden wären.

Im Weiteren ist die Frage zu klären, wann ein *Individuum* gut genug ist, um zu überleben. Ein naheliegendes Auswahlverfahren ist die *Bestenselektion*, bei der die besten Individuen ausgewählt werden, um zukünftige Eltern zu werden. Dieses Verfahren ist sehr einfach zu implementieren und findet auch häufig Anwendung. Eine weitere Möglichkeit ist die *q-stufige Turnierselektion*, bei der jedes *Individuum* mit $q$ zufällig ausgesuchten Individuen verglichen wird. Für jeden Vergleich, bei dem das *Individuum* besser ist als sein zufällig ausgewählter „Gegner“, erhält es einen Punkt. Zum Schluss werden die Individuen selektiert, welche die meisten Punkte erhalten haben. Der Aufwand dieses Verfahrens steigt mit wachsendem $q$. Auch kann dieses Verfahren zu einer „weicheren“ Selektion führen, da ein schwächeres *Individuum* auch überleben kann, wenn dessen Gegner noch schwächer waren.

Als letztes ist zu klären, wie lange dieses Optimierungsverfahren dauern soll. Dies wird durch ein zuvor definiertes Abbruchkriterium festgelegt. Hierzu kann man die Anzahl der Generationszyklen beschränken oder die Rechenzeit in Minuten. Ebenfalls bieten sich qualitative Kriterien an, wie das Erreichen einer gewissen Mindestgüte einer gefundenen Lösung, oder ein mögliches Stagnieren (= Konvergenz) der Qualität der *Population*. Letzteres kann so realisiert werden, dass ein Abbruch erfolgt, wenn innerhalb von $k$ Generationen keine Verbesserung erreicht wird. Natürlich kann man es auch dem Benutzer selbst ermöglichen, den Optimierungsprozess durch ein externes Abbruchsignal vorzeitig zu beenden.

Zusammenfassend lässt sich der Aufbau eines EA wie folgt darstellen:

- *Initialisierung* einer Population
- *Evaluierung* der Individuen mit Zielfunktion
- solange *Abbruchkriterium* nicht erfüllt ist, starte Generationszyklus:
  - ◇ *Selektion* zur *Reproduktion* (Elternindividuenauswahl)
  - ◇ *Reproduktion* durch Variation (Mutation/Rekombination)
  - ◇ *Evaluierung* der Kinder
  - ◇ *Selektion* der *Überlebenden* (Schrumpfen der Population)
- *Ausgabe* des besten Individuums

## 2.2 Künstliche Neuronale Netze

Künstliche Neuronale Netze (KNN) basieren auf den Erkenntnissen über die Funktionsweise des menschlichen Gehirns und bilden grundlegende Strukturen aus selbigen in abstrahierter Form nach.

Ein Neuron (Nervenzelle) kann man sich als eine Art Schaltelement vorstellen. Die Neuronen im Gehirn sind untereinander vernetzt durch Axonen, die die Verbindungsleitungen darstellen. Eine wichtige Funktion in diesem Netz haben dabei die Verbindungsstellen, die sogenannten Synapsen. Sie können grob in zwei Klassen eingeteilt werden, nämlich in anregende (afferent) oder hemmende (inhibitorisch). Unterschiedliche Synapsenstärken bestimmen nun, wie stark Elemente miteinander in Verbindung stehen und somit gemeinsam auf bestimmte Reize reagieren, worduch eine Logik entsteht. Die Stärken der Synapsen sind veränderbar (die sog. „synaptische Plastizität", siehe [Heb49]), wodurch erst ein Lernen möglich wird.

Für die Simulation solcher Netze auf Computern gibt es viele verschiedene Modelle für unterschiedliche Zwecke. Für unser Projekt kam jedoch nur eine Klasse zum Einsatz, nämlich sogenannte Feed-Forward-Netze.

### Feed-Forward Netze

Die Aufgabe eines solchen künstlichen neuronalen Netzes ist es, kurz gesagt, gegebene Eingabevektoren auf zugehörige Ausgabevektoren abzubilden. Bereits mit einem solchen relativ einfachen Netz lassen sich unterschiedliche Probleme lösen, wie z. B. Mustererkennung oder die Darstellung logischer Funktionen.

### Aufbau und Arbeitsweise

Ein Feed-Forward-Netz wird üblicherweise unterteilt in drei verschiedene Schichten: Die Eingangsschicht (input layer), die verborgenen Schicht (hidden layer), wobei hier auch durchaus mehrere möglich sind, und die Ausgabeschicht (output layer). Diese Schichten wiederum bestehen jeweils aus einer bestimmten Anzahl Neuronen. Jedes Neuron einer Schicht ist mit allen Neuronen der nachfolgenden Schicht verbunden (vollständig vermaschtes Netz). Diese Verbindungen haben Gewichte, die die Synapsen simulieren. Anregende bzw. hemmende Synapsen werden hier üblicherweise durch positive bzw. negative Werte simuliert. Die Eingangsleitungen zur ersten Schicht, sowie Ausgangsleitungen der Ausgabeschicht, besitzen hierbei keine Gewichte. Schließlich haben die einzelnen Neuronen noch eine (einheitliche) Aktivierungsfunktion. Gängige Funktionen sind z. B. die folgenden:

### Schwellenwertfunktion

$$f(x) = \begin{cases} 0, & \text{falls } x < \theta \\ 1, & \text{falls } x \geq \theta \end{cases}$$

für einen Schwellenwert $\theta$. Die Eingänge eines Neurons werden aufsummiert. Falls die Summe einen bestimmten Grenzwert (threshold), hier $\theta$, überschreitet, liefert die Funktion einen konstanten Wert, hier 1, als Ausgabe (das Neuron „feuert").

### Sigmoide Funktion (Fermifunktion)

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Die Eingänge eines Neurons werden wiederum aufsummiert. Die Summe dient als Eingabe für die Fermifunktion und der Funktionswert ist Ausgabe des Neurons. Dieser liegt zwischen 0 und 1.

Ein Feed-Forward-Netz arbeitet nun auf folgende Weise: An den Eingabeknoten des Netzes wird der Eingabevektor eingespeist. Diese Eingabe durchläuft nun entsprechend der Verbindungen sukzessive die einzelnen Schichten. Die Gewichte an den Verbindungen beinflussen dabei die Signale, indem sie mit ihnen multipliziert werden. An jedem Neuron einer Schicht werden diese Signale nun aufaddiert, durch die entsprechende Funktion ausgewertet und an die nächste Schicht weitergeleitet. Dieses Verfahren wird auch Propagation genannt.

#### Anlernen/Trainieren

Damit das künstliche neuronale Netz die Muster korrekt aufeinander abbildet, muss es trainiert werden, die Gewichte, die normalerweise mit Zufallswerten initialisiert werden, müssen also angepasst werden. Hier gibt es zwei häufig angewandte Verfahren. Ein Verfahren ist das sogenannte „backpropagation of error“ oder kurz einfach backpropagation. Bei diesem Verfahren wird, wie der Name bereits andeutet, der Fehler, berechnet aus der Differenz zwischen Soll- und Ist-Ausgabe, rückwärts durch die Schichten verbreitet und somit schrittweise behoben. Für jeden Ausgabeknoten wird also geprüft, ob sein tatsächlicher Ausgabewert größer, kleiner oder gleich dem gewünschten Ausgabewert ist. Ist er niedriger, so werden die Gewichte der Verbindungen, von allen Knoten der Vorgängerschicht zu diesem Knoten, herabgesetzt. Ist der Wert höher, werden sie erhöht. Im Falle der Gleichheit (bzw. innerhalb eines Toleranzbereiches) haben die Gewichte die richtigen Werte und müssen nicht angepasst werden. Dieses Vorgehen wird nun für alle Knoten Schicht für Schicht durchgeführt und schließlich mehrmals (evtl. auch sehr oft) wiederholt. Sind die Differenzen an den Knoten der Ausgabeschicht genügend klein (also innerhalb eines vorgegebenen Toleranzbereichs), kann der Trainingsvorgang abgebrochen werden. Mathemathisch entspricht dieses Vorgehen einem Gradientenabstieg. Dieser hat den Nachteil, dass man in einem lokalen Mininum „stecken bleiben“ kann, also die Fehlerfunktion somit nicht auf ein absolutes Minumum reduziert wird, und das so trainierte Netz nicht das Gewünschte leistet. In diesem Falle kann aber ein erneuter Trainingsversuch mit neu gewählten Synapsengewichten Abhilfe schaffen. Ein anderes gängiges Verfahren ist die Optimierung des Netzes durch einen Evolutionären Algorithmus, näher erläutert im Abschnitt 2.1.

## 2.3 Seminare (Abstracts)

### 2.3.1 Avatare und Kommunikation

Die Ausarbeitung in Anhang A auf Seite 71, beschäftigt sich mit Avataren und der Kommunikation in Computerspielen.

Bezüglich der Avatare wird darauf eingegangen, welche Funktion sie in Spielen einnehmen und vor allem, welche Rolle sie für den Spieler spielen. Hierbei wird erklärt, was man mit Avataren ausdrücken kann und welchen psychologischen Aspekt sie haben. Dies geschieht am Beispiel von “The Palace”, einer virtuellen Umgebung die auch als “Life Sim” bezeichnet wird.

Hinsichtlich der Kommunikation werden verschiedenen Möglichkeiten betrachtet, wie Spieler miteinander interagieren können, welche Funktionen die Kommunikation hat und worüber sich Spieler unterhalten. Neben diesen Schwerpunkten werden kurz angeschnitten, wie Chatbots versuchen menschlich zu wirken und wie man die Textgenerierung in Spielen individueller gestalten und an das Spielgeschehen und den Spieler anpassen kann.

### 2.3.2 Emotionen in der klassischen KI

Die Seminararbeit Emotionen in der klassischen KI, in Anhang B ab Seite 84 ausgeführt, liefert einen Überblick über das recht komplexe Forschungsgebiet. Anhand von Beispielen wird ein Einblick in die bisherige Forschung gegeben und somit die Arbeit motiviert. Mit Hilfe eines Emotionsmodells wird ein Regelsystem vorgestellt, welches situationsbewertend emotionale Zustände beschreibt.

### 2.3.3 Evolutionäre Algorithmen

Evolutionäre Algorithmen sind Zufallsheuristiken, welche die biologische Evolution als Optimierungsverfahren zum Vorbild haben.
Die Seminararbeit, Anhang C auf Seite 91, behandelt den formalen Hintergrund und stellt den strukturellen Aufbau von Evolutionären Algorithmen und ihrer einzelnen Module vor. Im vorletzten Kapitel wird ein Bezug zur Projektarbeit erstellt indem eine mögliche Anwendung in einem Strategiespiel vorgestellt wird.
Zum Abschluss werden noch allgemeine Anmerkungen zu Evolutionären Algorithmen gemacht - z.B. wann sich ihr Einsatz lohnt - sowie ihre Vor- und Nachteile aufgelistet.

### 2.3.4 Fuzzy Systeme

Die Fuzzy-Logik erweitert die Prädikaten-Logik, die nur zwischen zwei Zuständen unterscheidet, um einen Wertebereich dazwischen. Insbesondere die Fuzzy-Mengen ermöglichen die Zuordnung einer teilweisen Zugehörigkeit zu einer Menge. In einem Fuzzy-System werden dann Regeln auf diesen unscharfen Daten Folgerungen durchgeführt, bevor diese wieder in scharfe Signale umgewandelt werden. Es werden also für Eingabedaten die Zugehörigkeit zu bestimmten Gruppen bestimmt, die als Werte für ein statisches Folgerungssystem dienen, bevor dieses eine Konklusion als Ergebnis liefert.

Der geneigte Leser findet den ausführlichen Text in Anhang D ab Seite 101.

### 2.3.5 KI in den drei Spielen Poker, Diplomacy und Junta

Diese Ausarbeitung, Anhang E ab Seite 112, beschäftigt sich mit dem aktuellen Stand der Forschung zu den drei Spielen Poker, Diplomacy und Junta. Es wird untersucht welche Ansätze für den Entwurf der Computerspieler zur Zeit benutzt werden und die jeweiligen spielspezifischen Besonderheiten wie etwa Nichtdeterminismus in Form von zufälliger Kartenverteilung untersucht. Abschliessend werden die Spiele mit ihren Vor- und Nachteilen gegenübergestellt.

### 2.3.6 Ludologie

Die Ludologie ist ein wissenschaftlicher Forschungszweig, der sich mit kulturellen, strukturellen, kommunikativen und technischen Aspekten von Spielen, insbesondere digitalen Videospielen, auseinandersetzt. In dieser Ausarbeitung wollen wir zuerst verschiedene Typen von Spielen einteilen, um uns dann kurz um die mathematische Theorie von Spielen zu kümmern. In wenigen Worten werden wir dann erwähnen, wie die Entwicklung vom analogen zum digitalen Spiel stattgefunden hat. Im Hauptteil der Ausarbeitung werden wir uns mit verschiedenen Forschungsinhalten beschäftigen. Unter anderem werden wir uns die Geschichte des Spiels, die Definition des Spielbegriffs, Cybertext und ergodische Literatur, die Spielentwicklung, Immersion und Flow Theorie, persistente

Welten, Interfacedesign und auch Communities genauer ansehen. Die Seminararbeit ist in Anhang F auf Seite 120 zu finden.

### 2.3.7 Maschinelles Lernen

Die Seminararbeit, Anhang G auf Seite 130, klärt zu Beginn grundlegende Begriffe des Lernens sowie des logischen Schließens.

Dann wird konkret auf das Maschinelle Lernen eingegangen. Es wird die Unterteilung in das überwachte als auch das unüberwachte Lernen erläutert und Vertreter der einzelnen Gruppen jeweils angesprochen.

Anhand eines konkreten Beispiels werden grundlegende Probleme beim Maschinellen Lernen besprochen und wie man ihnen begegnen kann.

Abschließend werden Ideen für den Einsatz in Spielen gegeben.

### 2.3.8 Neuronale Netze

Vergleicht man die Arbeitsweise eines Computers mit der des menschlichen Gehirns, so lassen sich, ungeachtet der Unterschiede in der generellen Leistungsfähigkeit, deutliche Unterschiede feststellen. So ist das menschliche Gehirn beispielsweise tolerant sowohl gegenüber dem Ausfall einzelner „Rechenelemente“ als auch gegenüber unvollständigen Informationen. Beides sind Grundlagen für eine hohe Anpassungsfähigkeit. Künstliche Neuronale Netze versuchen nun, angelehnt an den Aufbau des menschlichen Gehirns (bzw. Teile davon), solche Fähigkeiten zumindest in grundlegender Form auf Computer zu übertragen. Hierbei gibt es eine Vielzahl verschiedener Modelle, die sich für viele unterschiedliche Zwecke einsetzen lassen.

Die Seminararbeit befindet sich in Anhang H auf Seite 142.

### 2.3.9 Soziologie/Psycholgie: Modelle für Emotionen/Verhalten von Spielern

Emotionen sind eine grundlegende Notwendigkeit für Spiele, da Spiele Reize auslösen, um Spiel-Verhalten und -Motivation zu beeinflussen. Ebenso sind Emotionen ein essentieller Bestandteil des menschlichen Charakters, sowie zwischenmenschliche Beziehungen für Interaktionen miteinander.

In der Seminararbeit „Psychologie/Soziologie: Modelle für Emotionen/Verhalten von Spielern“ Anhang I Seite 151, sollen die Einwirkungen dieser Faktoren auf das Denken und Handeln, sowie in diesem Kontext die Konnektivität zwischen Emotion und Verhalten, näher beleuchtet werden.

# 3 Erstes Projekt: Poker

## 3.1 Vorstellung der zur Wahl stehenden Computerspiele

Zu Beginn der Projektgruppe mussten sich die Teilnehmer entscheiden, welches Computerspiel Sie auswählen, um die geplanten Projektziele umzusetzen. Als Entscheidungshilfe hatten die Betreuer der Projektgruppe eine kleine Vorauswahl getroffen und somit den Kreis, der in Frage kommenden Spiele eingeschränkt. Zur Auswahl standen die Spiele Poker, Diplomacy und Junta.

Poker ist ein Kartenspiel, dass in der Regel mit zweiundfünfzig Karten gespielt wird. Die Karten besitzen verschiedene Wertigkeiten und lassen sich in vier unterschiedliche Symbole unterteilen (♣, ♡, ♠, ♢ - deutsche Bezeichnung für die vier Symbole sind Kreuz, Herz, Pik und Karo). Es geht darum aus insgesamt sieben Karten (fünf Gemeinschaftskarten und zwei Karten die jeder Spieler nur für sich hat) eine sogenannte „Hand“ zusammenzusetzen. Dabei werden die fünf geeignetsten Karten ausgewählt und bilden die bestmögliche Hand.

Es wird zwischen einer Vielzahl von Pokervarianten unterschieden. Die derzeit wohl beliebteste Pokervariante ist Texas Hold'em. Auf Grund seines schnellen Spielablaufs und einfacher Spielregeln hat das Spiel viele Anhänger gewonnen.
Das angestrebte Ziel ist es möglichst viel Geld (in Form von Geldscheinen) bzw. Chips von den Gegenspielern zu gewinnen. Dabei ist der Wert eines Chips gleichzusetzen mit dem Wert eines Geldscheines. Ein Spieler kann einen bestimmeten Geldbetrag auf sein Blatt (eigene Karten, die zu einer Hand kompletiert werden müssen) setzen, auf den seine Gegenspieler reagieren müssen, indem sie mit dem eingesetzten Betrag mitgehen (engl. *call*), erhöhen (engl. *raise*) oder ihr eigenes Blatt verwerfen (engl. *fold*). Poker wird in der Regel in mehreren Wettrunden gespielt (bei Hold'em vier). Die gesamten Chips einer Spielrunde werden letztendlich demjenigen Spieler mit der stärksten Hand zugewiesen oder dem einzig Verbliebenen, wenn alle anderen Spieler nicht bereit sind, den von ihm vorgelegten Einsatz ebenfalls zu bringen. Für nähere Erläuterungen sei auf den Abschnitt 3.3 verwiesen.

Bei Diplomacy handelt es sich um ein Brettspiel, welches die Karte Europas zu Beginn des 19. Jahrhunderts und die zur damaligen Zeit existierenden Machtverhältnisse zeigt. Der Spieler muss sich, bevor das Spiel angefangen hat, für eine Großmacht (Deutsches Reich / Frankreich / Großbritannien / Italien / Östereich-Ungarn / Osmanisches Reich / Russland) entscheiden, mit der er das Spiel bestreiten will. Ziel des Spiels ist es die Einflussbereiche so zu erweitern, dass insgesamt 18 von insgesamt 34 sogenannten Versorgungszentren (besonders gekennzeichnete Länder) eingenommen werden. Dabei ist es nötig in unbesetzte Länder einzudringen und sie zu besetzen oder bereits besetzte Länder anzugreifen, um sie zu erobern. Damit diese Ziele leichter umgesetzt werden können ist es möglich mit anderen Großmächten Allianzen zu schliessen, um sich gegen einen möglichen Angriff eines Gegenspielers zu schützen oder um die Ländergrenzen so zu erweitern, dass keine Großmacht in die eigenen Ländereien eindringen kann. An diese Absprachen müssen sich die einzelnen Parteien nicht unbedingt halten. Bei der für uns interessanten Computerspiel-Variante von Diplomacy werden alle wichtigen Komponenten, z.B. das Spielfeld oder die Bündnisse, auf einem Monitor visualisiert.

Das letzte Spiel, dass zur Auswahl stand, war Junta. Junta ist ein vom Pegasus Press vertriebenes Brettspiel, bei dem die Spieler die Rolle von Mitgliedern einer Militärjunta [1] in der sogenannten „Republica de las Bananas“ übernehmen. Am Anfang des Spiels

[1] Eine durch Staatsstreich an die Macht gekommene Gruppe von Offizieren, die die Regierungsgewalt diktatorisch ausüben.

muss ein Präsident gewählt werden, der seinerseits sechs Minister ernennt und zusammen mit diesen die Staatsleitung übernimmt. Der Präsident erhält bei seiner Ernennung eine Entwicklungsförderung aus dem Ausland und hat die Aufgabe seine Minister bei Laune zu halten und so zu versuchen einen möglichen Putsch zu verhindern. Die Minister ihrerseits wollen entsprechend entlohnt werden. Wie hoch das Gehalt der Minister ist, hängt vom Präsidenten ab. Wenn die Minister mit ihrem Einkommen oder der Regierungsführung nicht zufrieden sind, können Sie sich zusammenschliessen und einen Putsch starten. Falls es zu einem Putsch kommt und der Präsident gestürzt wird, muss ein neuer Präsident gewählt werden. Ziel des Spiels ist es, soviel Geld wie möglich auf einem Schweizer Bankkonto zu sammeln.

## 3.2 Entscheidung für ein Computerspiel und die entsprechende Plattform

Die Projektgruppe hatte sich zu einem frühen Zeitpunkt des Projekts für ein bestimmtes Computerspiel entschieden. Für die Evaluierung spielten mehrere Aspekte eine wesentliche Rolle. Zum einen war es den Teilnehmern der Projektgruppe wichtig, dass das von Ihnen ausgewählte Spiel die Möglichkeit bietet, Kommunikationstechniken leicht in den Kontext einzubetten. Zum anderen mussten die Spielregeln nicht zu komplex sein, so dass unerfahrene Spieler das Spiel in kurzer Zeit erlernen konnten. Natürlich spielten auch andere Punkte wie Spielspass und Spieltiefe eine Rolle bei der Entscheidungsfindung für ein konkretes Spiel. Die Tatsache, dass die PG davon ausging, dass ein vorhander Quellcode lediglich adaptiert werden soll, jedoch für das Spiel Junta bislang keine Open Source-Implementierung zur Verfügung stand, war ein wesentliches Argument gegen die Implementierung von Junta. Letztlich haben sich die Teilnehmer der Projektgruppe bei einer Abstimmung für das Spiel Poker entschieden.

Anschließend musste eine Entscheidung darüber getroffen werden, ob ein bereits fertiges Spiel erweitert werden soll oder auf Basis einer Pokerengine ein neues Spiel entwickelt werden soll. Da die Wahl an frei verfügbaren Poker-Plattform sehr groß war, kam die Projektgruppe sehr schnell zu dem Entschluss, dass eine Open-Source-Poker-Plattform verwendet werden soll. Um die Evaluierung voranzutreiben wurden zunächst mehrere Vorschläge für Open-Source-Poker-Plattformen in unserem Wiki[2] präsentiert und nach bestimmten Gesichtspunkten validiert. So sollte die verwendete Programmiersprache C/C++ oder Java sein und der Quellcode angemessen kommentiert vorliegen. Eine ansprechende graphische Benutzeroberfläche und vor allem ein zur Laufzeit fehlerfrei funktionierendes Programm waren ebenfalls wichtige Kriterien.

**PokerTH** Hierbei handelt es sich um ein in C++ implementiertes Pokerspiel, gleichwohl plattformunabhängig, weil auf QT[3] basierend. PokerTH ist netzwerkfähig. Avatare und eine textbasierte Kommunikation werden unterstützt. Der Quellcode ist nur spärlich kommentiert, wobei Kommentare sowohl in deutscher als auch in englischer Sprache auftreten.

**Netpoker** ist in C++ programmiert und weist eine auf den ersten Blick übersichtliche Klassenstruktur auf. Der Quellcode ist zwar kommentiert, aber leider sind die Kommentare in schwedischer Sprache verfasst worden. Das Spiel ist netzwerkfähig, jedoch unterstützt die aktuelle Version keine Avatare und bietet keine Chat-Funktion an.

**pokersource (Framework)** Ein in Python geschriebenes Framework, was jedoch in C bzw. C++ integriert werden kann. Es liegt keine Dokumentation vor. pokersour-

[2] http://de.wikipedia.org/wiki/Wiki
[3] http://trolltech.com/

ce stellt eine Netzwerkunterstützung bereit. Der Quellcode liegt in Paketen vor. Leider wird keine GUI und keine Chat-Funktionalität angeboten.

**PokerApp** Bei dieser Poker-Plattform handelt es sich um ein in Java geschriebenes Programm, welches netzwerkfähig und umfangreich dokumentiert ist. PokerApp läuft äusserst stabil und ist in mehreren Varianten verfügbar. Avatare werden nicht unterstützt. Bedauerlicherweise liegt nur eine rudimentäre Chat-Funktion vor.

**cspoker** In Java geschriebenes Pokerspiel, was ausreichend kommentiert vorliegt, aber keine Chat-Funktion und keine Avatare unterstützt.

**BotsPokerRoom** Es handelt sich bei BotsPokerRoom um ein in Java geschriebenes Pokerprogramm, wobei kein dokumentierter Code und keine graphische Oberfläche vorliegen.

Von den präsentierten Poker-Plattformen konnte PokerTH unseren Anforderungen am ehesten gerecht werden. Ein entscheidendes Kriterium hierfür war, dass PokerTH als Open-Source-Poker-Plattform sehr weit verbreitet ist, so dass bei einer möglichen Veröffentlichung ein breites Publikum unsere Ergebnisse begutachten kann und eventuell, falls die Bereitschaft besteht, Bewertungen hinsichtlich unserer Erweiterungen abgeben kann. Ein weiteres Kriterium war die mit nicht allzu großem Aufwand durchzuführende Adaption auf unsere Bedürfnisse. Die zahlreichen Features (Chat-Funktion, Avatare, Netzwerk- und Sound-Engine) und die ansprechende graphische Oberfläche haben sich letztendlich zu Gunsten von PokerTH ausgewirkt.

## 3.3 PokerTH

PokerTH[4] ist der Name eines Open-Source-Software-Projektes. Es handelt sich um eine sehr beliebte und verbreitete Computersimulation für die Pokervariante Texas Hold'em. Die Software steht in all ihren Teilen unter der GNU General Public License (GPL). Es wurde in der Programmiersprache C++ unter Verwendung der Bibliotheken Qt4, Boost und SDL programmiert und ist für die Plattformen Windows, Linux und MacOS X verfügbar.

Die erste öffentliche Version war die Version 0.2. Viele heutzutage gängigen Features wurden in dieser Version noch nicht unterstützt. Die zum Start unseres Projektes aktuelle Version war 0.6.2. Bereits in vorherigen Versionen wurde eine Netzwerk- und Sound-Engine implementiert.

Um den Einstieg in die Spieleplattform PokerTH zu erleichtern, werden kurz die Spielregeln der Poker-Variante Texas Hold'em erläutert. Hold'em im Allgemeinen bezeichnet all jene Poker-Varianten, bei denen fünf offene Karten (engl. *Board cards* bzw. *Community cards*) in die Mitte des Tisches gelegt werden, die von jedem Spieler verwendet werden können, um eine Hand (die fünf besten Karten, die ein Spieler nutzen kann) zu bilden, während jeder Spieler jedoch nicht mehr als zwei seiner Hand-Karten (engl. *Hole cards*) verwenden darf. Bei den verwendeten Karten handelt es sich um französische bzw. anglo-amerikanische Karten zu 52 Blatt. Es können zwei bis maximal vierzehn Personen an einem Spiel teilnehmen. Ziel ist es, die höchste Poker-Kombination zu erhalten oder durch geschickte Spielweise die anderen Spieler zur Aufgabe zu bewegen.

Die Rangfolge der einzelnen Kartenkombinationen ist wie folgt festgelegt, beginnend mit dem besten Blatt:

1. **Royal Flush** (z.B.: 10♣, B♣, D♣, K♣, A♣): Zehn bis Ass der gleichen Farbe.

[4] `http://www.pokerth.net/`

2. **Straight Flush** (z.B.: 2♡, 3♡, 4♡, 5♡, 6♡): Fünf aufeinander folgende Karten in der gleichen Farbe.
3. **Four of a Kind** (z.B.: A♣, A♠, A♡, A♢): Vier Karten mit dem gleichen Zahlen- oder Bildwert.
4. **Full House** (z.B.: 10♠, 10♢, A♡, A♠, A♣): Drei Karten des gleichen Zahlen- oder Bildwertes und zwei andere Karten mit dem gleichen Zahlen- oder Bildwert.
5. **Flush** (z.B.: 3♠, 5♠, 9♠, 10♠, Q♠): Fünf Karten der gleichen Farbe.
6. **Straight** (z.B.: 4♡, 5♠, 6♡, 7♣, 8♢): Fünf aufeinander folgende Karten, egal in welcher Farbe.
7. **Three of a Kind** (z.B.: 7♢, 7♣, 7♠): Drei Karten mit dem gleichen Zahlen- oder Bildwert.
8. **Two Pair** (z.B.: 7♢, 7♣, A♢, A♡): Zweimal zwei Karten mit dem gleichen Zahlen- oder Bildwert.
9. **One Pair** (z.B.: 7♢, 7♣): Zwei Karten mit dem gleichen Zahlen- oder Bildwert.
10. **High Card** (z.B. A♠): Einfach eine hohe Karte, sonst nichts.

Zu Beginn muss ein sogenannter Kartengeber (engl. *dealer*) bestimmt werden. Hierzu wird der Spieler mit der höchsten Karte als *dealer* bestimmt und erhält eine Markierung zur Kenntlichmachung (engl. *dealer button*). Die Rolle des Kartengebers wechselt immer nach jedem einzelnen Spiel im Uhrzeigersinn. Der Spieler zur Linken des Gebers muss einen vorgeschriebenen Einsatz (engl. *small blind*), zum Beispiel zehn Geldeinheiten, setzen, sein Nachbar einen erhöhten *small blind*, den sogenannten *big blind* (meistens der doppelte *small blind*). Die übrigen Spieler müssen vor Erhalt der ersten Karten keinen Grundeinsatz leisten. Das Spiel verläuft in maximal vier Wettrunden. Jeder Teilnehmer erhält eine Starthand bestehend aus zwei verdeckten Karten (engl. *hole cards*). Nun müssen die Spieler im Uhrzeigersinn Ihre Einsätze machen. Dabei ist es möglich den momentan höchsten Einsatz am Tisch mitzugehen (engl. *call*), den aktuell höchsten Einsatz der Runde weiter zu erhöhen (engl. *raise*), was als *bet* bezeichnet wird, wenn es sich um die erste Erhöhung der Runde handelt, oder seine beiden *hole cards* zu verwerfen (engl. *fold*). Wenn kein Spieler mehr erhöhen will und jeder den gleichen Einsatz erbracht hat, dann gilt die erste Wettrunde als beendet (engl. *flop*), nachdem der Kartengeber drei Gemeinschaftskarten aufgedeckt hat. Es folgt die zweite Wettrunde (engl. *turn*). Diese Runde verläuft analog zur vorherigen, nur das am Ende der Runde nicht drei Karten in der Mitte des Spielfeldes aufgedeckt werden, sondern eine. Nach einer weiteren Wettrunde (engl. *river*) werden die Gemeinschaftskarten komplettiert und sind jetzt fünf an der Zahl. Es folgt eine abschliessende Wettrunde. Alle Spieler, die in den vorherigen Runden ihr Blatt nicht verworfen haben, können (wenn sie der Meinung sind mit den Karten den Pot zu erlangen) nach dieser Runde ihre beiden verdeckten Handkarten den anderen Spielern zeigen, um den endgültigen Sieger dieser Runde zu bestimmen. Derjenige mit dem besten Blatt gewinnt alle Einsätze, die in dieser Runde gemacht wurden.

Das in Abbildung 1 abbgebildete Bildschirmfoto stellt die die grafische Benutzeroberfläche von PokerTH dar. Die bei Markierung 1 abbgebildete Spielkarte bezeichnet die erste *Hole-Card* (erste eigene Karte). Markierung 2 die zweite *Hole-Card*. Mit dem Knopf bei Kennzeichnung 3 kann der Einsatz erhöht werden bzw. ein Einsatz geboten werden, wenn es sich um den ersten Einsatz einer Runde handelt. Um diese Option zu nutzen kann bei 6 ein bestimmter Betrag, der eingesetzt werden soll, eingetragen werden oder mit Hilfe eines Schiebereglers ein Betrag festgelegt werden. Der Knopf kann auch als zweiter Einsatz (als *reraise*, sofern zuvor ein raise zu einem bet getätigt wurde) benutzt werden.


**Abbildung 1:** Bildschirmfoto einer Spielsituation aus der Computersimulation PokerTH

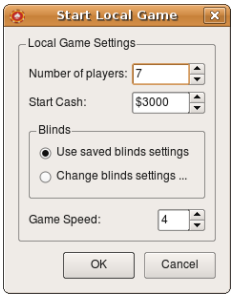



**Abbildung 2:** Fenster beim Starten des Spiels PokerTH

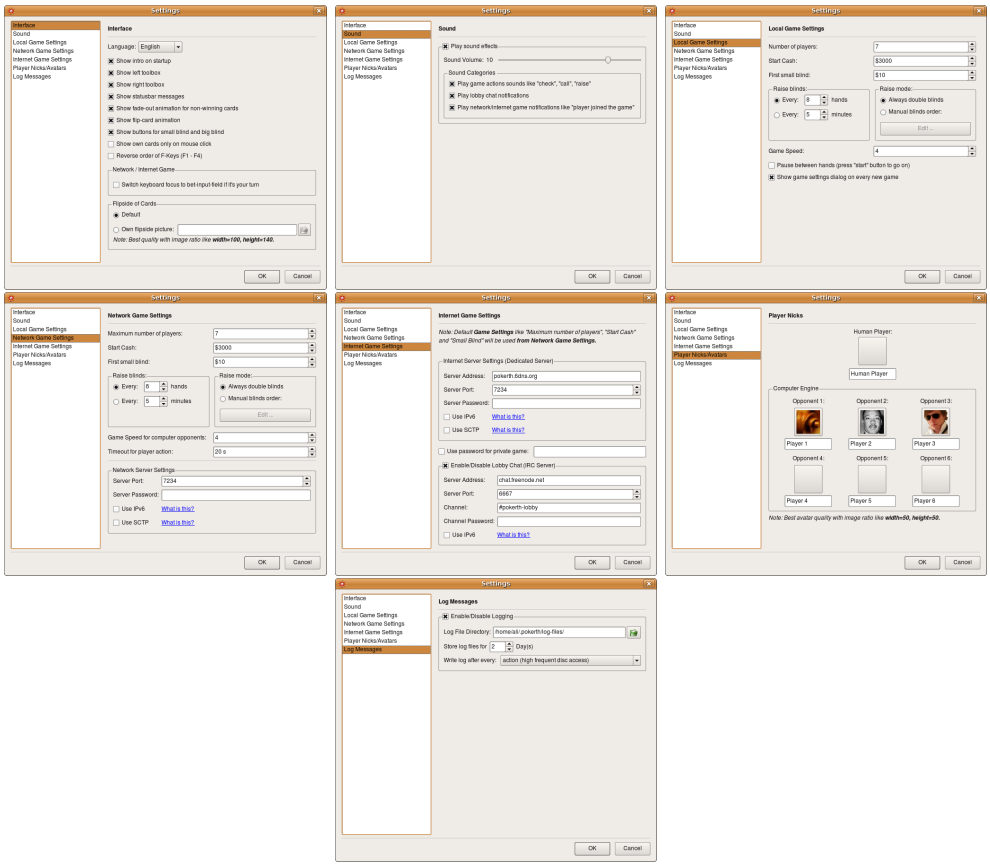


**Abbildung 3:** Bildschirmfotos des Menü-Punktes “Settings”

Mit dem Knopf bei Kennzeichnung 4 kann an den nächsten Spieler weitergegeben werden, falls der eigene Einsatz dem aktuellen (höchsten) Einsatz entspricht bzw. mitgegangen werden, wenn der eigene Einsatz niedriger ist als der aktuelle Einsatz. Mit dem Knopf 5 kann das aktuelle Blatt verworfen werden. Bei Markierung 7 wird angezeigt, ob es sich bei der aktuellen Position um die Position des *big blinds* bzw. *small blinds* oder *dealer buttons* handelt; Insofern nichts angezeigt, muss die aktuelle Position anhand des *dealer buttons* bestimmt werden 8 markiert die Stelle an der ein Avatar angezeigt werden kann. Hierzu muss über das Auswahl-Menü bei Markierung 16 ein entsprechender Avatar angegeben werden (eine detaillierte Beschreibung des Auswahl-Menüs erfolgt zu einem späteren Zeitpunkt). Über dem Avatar befindet sich ein in weisser Schrift gehaltener Schriftzug “Cash:”, darunter wird der eigene Geldbetrag (Währung US-Dollar ($)) angezeigt. An Position 9, 10 und 11 werden direkt nach Beendigung der ersten Wettrunde jeweils eine neue Karte aufgedeckt (*flop*). Eine zusätzliche Karte wird an Position 12 angezeigt, sobald die zweite Wettrunde beendet ist und eine weitere Karte erscheint an Position 13 nach Beendigung der dritten und vorletzten Wettrunde. Diese insgesamt fünf Karten, die sich in der Mitte des Spielfeldes befinden, bilden die *Gemeinschaftskarten*. 14 markiert den Punkt an dem die Wertigkeiten der einzelnen Blätter dargestellt werden. Die Spielgeschwindigkeit wird mithilfe eines Schiebereglers angepasst (Markierung 15). Das bei Kennzeichnung 15 dargestellte Informations-Fenster zeichnet den Spielverlauf mit, so dass alle vorangegangenen Spielsituationen eingesehen werden können. Markierung 16 und 17 sind zwei Menüpunkte, mit deren Hilfe sich diverse Optionen einstellen lassen. Über Markierung 16 wird ein lokales Spiel gestartet. Es erscheint ein neues Fenster (Abbildung 2), bei dem eine Reihe von Einstellungen getroffen werden

können. Es kann eine bestimmte Anzahl von Gegenspielern angegeben (von minimal einem bis zu maximal sieben; standartmässig auf vier gesetzt), das Anfangsgeld aller Spieler auf einen bestimmten Wert gesetzt (standartmässig auf $3000 gesetzt) und die Spielgeschwindigkeit festgelegt werden (von minimal eins bis zu maximal elf). Ausserdem ist es möglich über einen radio-button die *blinds* während des Spielverlaufs stetig zu erhöhen oder immer konstant bei den gleichen Werten zu belassen. Des Weiteren ist es möglich über Kennzeichnung 16 ein neues Netzwerk-Spiel zu initiieren oder einer bestehenden Netzwerk-Partie beizutreten. Der Menü-Punkt *Settings*, der über Markierung 16 erreicht werden kann, beinhaltet etliche Einstellungmöglichkeiten. So kann ein bestimmter Avatar ausgewählt oder die Sound-Engine angepasst werden. Abbildung 3 zeigt alle Untermenüpunkte des Menü-Punktes *Settings*. Ein weiterer Menüpunkt, der unter Markierung 16 erreicht werden kann, ist der Menü-Punkt *About PokerTH*. Dieser enthält Informationen über das Projekt, die Verantwortlichen und Lizenzen, ausserdem werden Danksagungen ausgesprochen. Der Menü-Punkt *Quit* beendet das Programm. Über Markierung 17 kann die Sicht auf das aktuelle Spielfeld verändert werden. So ist es möglich in den Vollbildschirm-Modus zu wechseln oder zusätzlich ein Chat-Fenster anzuzeigen, dass als neuer Tab bei Markierung 14 dargestellt wird.

## 3.4 Zeitplan

Massgeblich für unsere Zeiteinteilung waren die „Milestones“, die wir uns zu Beginn der Projektgruppe gesetzt haben. Abbildung 4 zeigt unsere angestrebten Milestones. Unser Zeitplan sah vor, dass zunächst in einer sogenannten Planungsphase die Ziele

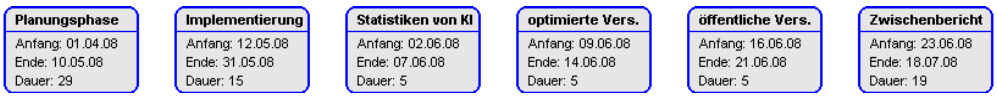



**Abbildung 4:** Die Projektplanung zu PokerTH. Die Grafik zeigt unseren ersten Entwurf eines Zeitplans des Projekts

der einzelnen Teilnehmer genauer definiert werden und anhand dieser sich bestimmte Aufgabenstellungen ergeben sollten. Differenziertere Konzepte sollten in Untergruppen erarbeitet und immer weiter verfeinert werden. Anschliessend mussten die entwickelten Konzepte umgesetzt werden. Dies sollte in der Implementierungsphase stattfinden. Nachdem eine erste lauffähige Version implementiert wurde, soll der nächste Abschnitt, den wir unter der Überschrift „Statistiken für KI“ zusammengefasst haben, eine Feldstudie mit ca. 50 Probanden beinhalten, wobei die gesammelten Daten aufgezeichnet werden sollten. Nach sorgfältiger Validierung sollte die aktuelle Version, soweit adaptiert werden, dass eine optimierte Version entsteht, die vorherige Fehler behebt und suboptimale Algorithmen verbessert. Diese optimierte Version sollte nach einem erneuten Test zu einer endgültigen Version führen, die der Öffentlichkeit zugänglich gemacht wird. Abschliessend sollte ein Zwischenbericht unsere Arbeit des Semesters zusammenfassend wiedergeben.

## 3.5 Charakter und Emotionen

Um das Kernziel der Projektgruppe, das Erzeugen eines menschenähnlichen Computerspielers, zu verwirklichen, sollten Emotionen ein essenzieller Bestandteil des Computerspielers werden. Diese Emotionen sollten sich, wie bei einem Menschen, durch Ereignisse während des Pokerspieles verändern und sich im Verhalten, insbesondere in der Spielweise, widerspiegeln. So sollte es möglich sein, dass wenn ein Computerspieler über

mehrere Hände hinweg immer verliert, dieser wütend wird, den Kopf verliert und seine Spielhandlungen völlig widersprüchlich werden. In der Pokerwelt gibt es hierfür sogar einen eigenen Begriff *on tilt*. Darüber hinaus war es ein Ziel, dass der Computerspieler einen individuellen Charakter erhält, welchen man ebenfalls an seinem Spielverhalten erkennen können sollte.

### 3.5.1 Charakterwerte

Am Anfang stand die Entscheidung über die Charakterwerte, die den Charakter eines Computerspielers beschreiben bzw. codieren. Es wurden verschiedene Begriffe wie *Aggressivität* und *Risikofreude* gesammelt. Anschließend wurde darüber diskutiert, was diese eigentlich bedeuten und wie sie die Spielweise eines Menschen beim Poker beeinflussen. Da es viele Überschneidungen in den Bedeutungen gab, wurden einige Begriffe zusammengefasst. Zum Schluss haben sich folgende Charakterwerte herauskristallisiert, welche in der Implementierung als Integer zwischen 0 und 100 dargestellt werden. Sie werden in der Datei *config.xml* (vgl. Seite 43) definiert, bleiben während des Spieles konstant und sind für die Berechnung der Emotionen wichtig.

**Setzverhalten Geld** gibt an, wie groß die Bereitschaft ist, sein Geld einzusetzen. Ist dieser Wert gering, wird höchstens der Mindestbetrag eingesetzt (checken oder callen). Je höher dieser Wert ist, desto höhere Beträge raist der Spieler.

**Spielverhalten der Hände** legt fest, welche Hände der Spieler spielt. Bei einem niedrigen Wert werden nur wirklich gute Hände gespielt, während bei einem hohem Wert nur selten gefoldet wird.

**Karten bewerten** codiert die Fähigkeit des Spielers, seine Karten hinsichtlich Verbesserung einzuschätzen. Je höher dieser Wert ist, desto besser kann der Spieler die Chancen eine Karte zu erhalten, die seine bisherige Kartenkombination verbessert, einschätzen.

**Gedächtnis** ist ein Prozentwert. Er gibt an, wieviel Prozent der eigenen und gegnerischen Hände von maximal 500 Runden sich der Spieler merken kann. Die 500 Runden wurden Aufgrund des Spieles gegen den Pokercomputer Polaris [5] als maximal zu spielende Anzahl Hände festgelegt.

**Wartezeit** legt fest, wie lange der Spieler benötigt, um seinen nächsten Schritt durchzudenken bevor er ihn ausführt. Ursprünglich sollte das auch im Spiel simuliert werden. Die Computerspieler sollten merklich unterschiedlich lange nachdenken, bevor sie eine Handlung ausführen. Diese Eigenschaft wurde aber aus Zeitgründen nicht implementiert.

**Gegner einschätzen** gibt an, wie gut der Spieler seine Gegenspieler, deren Verhalten und ihre Karten einschätzen kann. Er ist für das *Opponent Modeling* (siehe Kapitel 3.7 auf Seite 33) von besonderer Bedeutung.

**Anpassungsfähigkeit** sagt aus, wie schnell sich der Spieler auf die Spielweise des Tisches, und somit auf die Spielweise seiner Gegner, einstellen kann. Somit wird auch indirekt angegeben, wie gut sich der Spieler an das Spiel seiner Gegner anpassen kann.

**Blatt Wert** gibt an, wie gut der Spieler die Blätter und ihren Wert einschätzen kann. Ein geringer Wert bedeutet, dass der Spieler z.B. ein doppeltes Paar für höher als einen Drilling hält. Dadurch wird indirekt codiert, wie gut der Spieler mit dem Spiel und seinen Spielregeln vertraut ist. Ein geringer Wert steht somit für einen Anfänger.

[5] `http://www.cs.ualberta.ca/~games/poker/man-machine/` [21.07.2008]

Zu einem späteren Zeitpunkt des Projektes wurden noch weitere Charakterwerte eingeführt, die keine Auswirkungen auf die Emotionen, wie die obigen, haben, sondern einen besondere Rolle für die Kommunikation (vgl. Abschnitt 3.5.5) spielen. Auch diese Werte werden mittels Integer zwischen 0 und 100 implementiert, in der Datei *config.xml* (vgl. Seite 43) definiert und bleiben während des Spieles konstant.

**Ehrlichkeit** beschreibt, ob ein Spieler viel blufft (geringer Wert) oder ehrlich zu seinen Mitspielern ist (hoher Wert). In der Implementierung wurde dieser Wert zu Gunsten einer einfacheren Implementierung außer acht gelassen, so dass die Bots immer ein Emoticon anzeigen, welches ihrer stärksten Emotion entspricht.

**Kommunikativität** gibt an, wieviel und wie häufig der Spieler mit den anderen Spielern am Tisch kommuniziert. Je höher dieser Wert ist, desto mehr Nachrichten werden vom Bot verschickt.

**Persönlichkeitsausprägung** codiert, wie introvertiert (geringer Wert) oder extrovertiert (hoher Wert) ein Spieler ist. Dieser Wert wurde in der späteren Implementierung jedoch nicht benötigt.

### 3.5.2 Emotionsmodell

Der nächste Schritt bestand darin, sich über die Emotionen des Computerspielers Gedanken zu machen. Hierbei konnte auf die Seminararbeit über Modelle für Emotionen (siehe 151) zurück gegriffen werden. Nach einer Internetrecherche stellten sich das *OCC-Modell* und das *Arousal-Valence Modell* als besonders weit verbreitet heraus. Das *OCC-Modell* erschien für das Vorhaben eines emotionalen Pokerspielers allerdings zu komplex und aufwendig zu implementieren. Bei dem *Arousal-Valence Modell* wiederum war nicht klar, wie sich die dort codierten Werte im Pokerspiel wiederfinden.
Wie auch Wenzel Svojanovsky in seiner Diplomarbeit ([Svo06], Abschnitt 3.2.1.4), hat sich die Projektgruppe schließlich für ein eigenes, eher einfaches Emotionsmodell entschieden. Es wurden verschiedene Emotionen gesammelt um dann die, die sich ähnlich sind, zu Gruppen zusammenzufassen:

1. Wut / Ärger / Hass
2. Euphorie / Glück / Freude / Schadenfreude
3. Nervosität / Angst / Panik / Besorgnis
4. Selbstsicherheit
5. Trauer

Bei der Betrachtung dieser Gruppen fällt auf, dass die Gruppe 4 einen Gegenpol zur Gruppe 3 und die Gruppe 5 einen Gegensatz zur Gruppe 2 darstellt. Es wurden also auch diese Gruppen zusammengefasst und innerhalb jeder Gruppe eine Ordnung definiert. In der Implementierung erhielt jede Gruppe einen ganzzahligen Wertebereich zwischen 0 und 100.

**Neutral, Ärger, Wut, Hass** hat ihren neutralen Punkt bei 0. Besonders ist hierbei, dass dieser Wert nicht nur einmal global existiert, sondern auch für jeden Gegenspieler als gerichteter Emotionswert. Der globale Wert ergibt sich aus dem Durchschnitt aller gerichteten Werte.

**Trauer, Freude, Glück, Euphorie** hat ihren neutralen Punkt bei 50.

**Selbstsicherheit, Neutralität, Besorgnis, Nervosität, Angst, Panik** hat ihren neutralen Punkt bei 50.

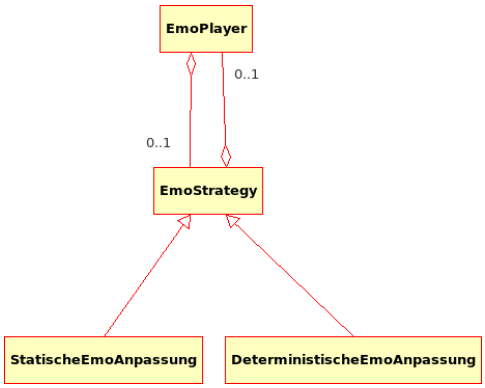



**Abbildung 5:** Zur Emotionsgenerierung und -veränderung wurde ein *Strategey Pattern* benutzt

Es wurde auch angedacht in die erste Gruppe den Gegenpol *Liebe* aufzunehmen. Bei der *TV Total PokerStars Nacht*[6] konnte man beobachten, wie Sympathie für einen Gegenspieler das Spielverhalten beeinflussen kann. Dazu gehörte das absichtliche Verlieren von Elton, damit Stefan Chips erhält, und das weniger aggressive Vorgehen gegen die einzige weibliche Pokerspielerin. Doch es wurde vermutet, dass dieses Verhalten stark durch die zuschauenden Kameras motiviert wurde. Bei einem Computerspiel hingegen, bei dem es keine Notwendigkeit gibt, sich vor Außenstehenden zu repräsentieren, wird sehr wahrscheinlich niemanden zuvorkommender behandeln als jemand anderes. Darüber hinaus dürfte das Verhalten, wenn man starke Zuneigung empfindet, ähnlich kopflos wirken, wie wenn man besonders wütend ist. Aus diesen Gründen wurde diese Emotion nicht in das Modell aufgenommen.

Um die Emotionen zu generieren und während des Spieles zu verändern, wurde ein *Strategy Pattern* implementiert (vgl. Abbildung 5).

Dieses *Design Pattern* hat den Vorteil, dass es später sehr einfach ist, andere Ideen zur Generierung und Veränderung der Emotionen zu integrieren und verschiedenen Strategien zu benutzten. Welche Strategie von einem Bot genutzt wird, kann in der Datei *config.xml* (vgl. Seite 43) eingestellt werden. Neben den implementierten Strategien, die später beschrieben werden, standen noch Ideen im Raum, die Generierung und Veränderung durch ein Neuronales Netz zu bewerkstelligen. Außerdem erschien die *EmoMap* von Wenzel Svojanovsky ([Svo06], Abschnitt 3.2.1) interessant und vielversprechend. Eine *EmoMap* kann man sich als eine Landschaft mit Bergen und Tälern vorstellen, in der eine Kugel (von Wenzel Svojanovsky als Zeiger bezeichnet) gemäß den physikalischen Gesetzen Reibung, Schwerkraft und Zeit umher rollt. Bestimmte Bereiche werden Emotionen zu geordnet. Wenn sich die Kugel in einem Bereich befindet, hätte der Computerspieler diese Emotion. Durch etwaige Ereignisse würde die Kugel einen Stoß erhalten und ihre Position oder ihre Bewegung verändern. Leider konnten diese Ideen aus Zeitgründen nicht umgesetzt werden.

### 3.5.3 Generierung der Emotionen

Vor Beginn des Spieles, wenn die Computerspieler erstellt werden, werden die Emotionswerte anhand der Charakterwerte berechnet. Die Charakterwerte können vor dem Starten des Programms in der Datei *config.xml* (vgl. Abschnitt 3.8.2 auf Seite 43) definiert werden. Im Weiteren wird *Wut* stellvertretend für die Emotionsgruppe 1, *Freude* für die Emotionsgruppe 2 und *Nervosität* für die Emotionsgruppe 3 verwendet (für die

[6] `http://tvtotal.prosieben.de/show/letzte_sendung/2008/05/01/12083.html` [22.07.08]

Bedeutung der Gruppen siehe Seite 16).
Wie in Abbildung 5 zu erkennen ist, wurden zwei konkrete Emotionsstrategien implementiert. Die Klasse *StatischeEmoAnpassung* kommt einem Dummy nahe. Sie setzt alle Emotionen bei der Erstellung des Computerspielers auf den neutralen Wert.

**Wut** = 0. (Dies gilt sowohl für den globalen sowie für alle gerichteten Wutwerte.)

**Freude** = 50.

**Nervosität** = 50.

In der Klasse *DeterministischeEmoAnpassung* wurde die eigentliche Emotionsgenerierung implementiert. Da es zu diesem Thema keine Referenzliteratur gibt, wurde für jede Emotion überlegt, ob sich ein bestimmter Charakterwert positiv oder negativ auf sie auswirkt. Wenn man weiß, dass man ein gutes Gedächtnis hat, man die Karten sehr gut bewerten und seine Gegner sehr gut einschätzen kann, dann wird dieses die *Wut* nicht steigern. Je besser man sich anpassen kann, also auch je schneller man sich an den Raum, in dem gespielt wird, gewöhnt, desto stärker wird eine eventuell vorhandene *Wut* gesenkt. Die Generierung des Wutwertes ist bewusst darauf ausgelegt, dass sie zu Beginn nicht 0 ist, weil man seine Gegner besiegen möchte und somit mit einer gewissen Grundaggressivität ins Spiel startet.
Ähnlich verhält es sich bei *Freude*. Es ist sehr unwahrscheinlich, dass man am Anfang eines Pokerspieles traurig ist, weswegen alle Charakterwerte positiv eingehen. Auch hier gilt, desto besser ich die Karten bewerte und meine Gegner einschätzen kann, desto mehr freue ich mich auf ein Spiel.
Bei *Nervosität* können sich die Charakterwerte sowohl positiv wie auch negativ auswirken. Dies liegt daran, dass man zu Beginn sowohl selbstsicher als auch nervös sein kann. Hat man ein gutes Gedächtnis, kennt die Spielregen, kann die Karten gut bewerten und die Gegner gut einschätzen, wird man zu Beginn eher selbstsicher sein. Hier gilt auch die Umkehrung. Wenn man ein schlechtes Gedächtnis hat, die Spielregen und Kartenwerte nicht kennt, die Karten nicht gut bewerten und seine Gegner nicht gut einschätzen kann, wird man wahrscheinlich nervös sein. Die *Wartezeit* wurde an dieser Stelle mit der Überlegung aufgenommen, dass wenn eine Person lange für seine Entscheidungen braucht, diese sehr gut durchdacht sind und die Person somit nicht nervös ist. Eine schnelle Handlung deutet eher auf eine Kurzschlussreaktion hin.

**Wut** = (100 - *Karten bewerten*) * 0,1
+ (100 - *Gedächtnis*) * 0,05
+ (100 - *Gegner einschätzen*) * 0,1
+ *Anpassungsfähigkeit* * -0,05

**Freude** = 50
+ *Karten bewerten* * 0,05
+ *Gedächtnis* * 0,05
+ *Gegner einschätzen* * 0,05

**Nervosität** = 50
+ (50 - *Karten bewerten*) * 0,1
+ (50 - *Gedächtnis*) * 0.05
+ (75 - *Wartezeit*) * 0,15
+ (50 - *Gegner einschätzen*) * 0,1
+ (50 - *Anpassungsfähigkeit*) * 0,15
+ (50 - *Blatt Wert*) * 0,02

Alle Emotionswerte können maximal 100 und minimal 0 sein, wie auch die Charakterwerte. Sollte ein Emotionswert bei der Berechnung aus diesem Bereich hinaus laufen, wird er automatisch angepasst.
Vermutlich wird diese Berechnung noch viel Verbesserungspotential besitzen. Bei der

Erstellung dieses Modells bestand die Hoffnung, dass es genug Zeit gibt, dieses am Ende des Projektes entsprechend zu validieren und zu verbessern. Was zum Schluss leider nicht der Fall war.

### 3.5.4 Veränderung der Emotionen

Wie bei der Generierung stellt die Klasse *StatischeEmoAnpassung* eine Art Dummy dar. Wird diese Klasse zur Emotionsanpassung verwendet, bleiben die Emotionen immer konstant und verändern sich nicht.
Ebenso findet die eigentliche Emotionsveränderung in der Klasse *DeterministischeEmoAnpassung* statt. Zu jedem Ereignis, das Einfluss auf Emotionen nimmt, gibt es eine eigene Methode. Wie die Emotionen verändert werden, wurde mangels wissenschaftlicher Literatur zu diesem Kontext dadurch bestimmt, indem sich die für die Emotionen verantwortlichen Mitglieder der Projektgruppe gefragt haben, wie sie sich selber in der jeweiligen Situation fühlen. Ähnlich wie bei der Generierung der Emotionen wird dieses Modell noch Verbesserungspotential besitzen. Für eine gezielte Validierung und anschließende Überarbeitung des Berechnungsmodells fehlte leider die Zeit.
Der Code zu den einzelnen Methoden befindet sich im Anhang N.1 auf Seite 168.

#### Emotionsveränderungen bei gegnerischen Aktionen

Generell kann gesagt werden, dass sich die Emotionen bei gegnerischen Aktionen eher wenig ändern. Es besteht die Möglichkeit, dass ein Spiel mit 6 Gegenspielern beginnt. Da sich die Emotionen in diesem Fall sechs mal pro Runde anpassen, ist es nicht sinnvoll an dieser Stelle sehr große Veränderungen herbeizuführen. Die nächsten Abschnitte beschreiben, wie sich die Emotionen bei bestimmten gegnerischen Aktionen ändern. Für die exakten Werte sei auf den Anhang N.1 ab Seite 168 verwiesen.

**Gegner erhöht (raise)** Erhöht ein Gegenspieler seinen Einsatz, so verändern sich die Werte für *Wut*, *Freude* sowie *Angst*. Zuerst wird nach der Spielphase (Pre Flop, Flop, Turn, River) unterschieden. Je weiter ein Spieler mit einer Hand gekommen ist, je mehr Einsatz musste er leisten, um im Spiel zu bleiben. Aus diesem Grund verstärken sich die Emotionsveränderungen proportional mit der Phase des Spiels. Zusätzlich wird unterschieden, wie gut die Karten des Spielers sind.

***Wut*** Die Emotion *Wut* wird generell variabel erhöht, kann aber durch die Charaktereigenschaft *Gegner einschätzen* leicht gesenkt werden. Je nach Wert der Charaktereigenschaft *Gegner einschätzen* kann die Wut aber auch steigen. Genauso beeinflusst die *Anpassungsfähigkeit* die Wut. So ist auch der Wert der *Anpassungsfähigkeit* entscheidend für eine Erhöhung oder Verringerung der Wut.

***Freude*** Die Emotion *Freude* kann sich durch die Aktion *Gegner erhöht* sowohl verringern als auch steigern. Erhöht ein Gegner und der Spieler hat schlechte Karten, so verringert sich seine Freude. Der Spieler kann nur dann gewinnen, wenn sein Gegner schlechtere Karten hat, also blufft. Erhöht der Gegner aber mit besseren Karten, so ist ein Sieg des Spielers nur durch einen erfolgreichen Bluff möglich. Erhöht ein Gegner allerdings und der Spieler hat gute Karten, so steigt seine Freude, da er eine größere Chance hat, das eingesetzte Geld zu gewinnen. Die Freude wird außer von den Kartenwerten nur noch von der Charaktereigenschaft *Gegner einschätzen* beeinflusst.

***Angst*** Die Emotion *Angst* kann sowohl steigen als auch sinken, was wiederrum von den Karten als auch von der Spielphase abhängt. Je nach Phase und Kartenwert

sind die Charaktereigenschaften *Gegner einschätzen* und *Anpassungsfähigkeit* für Veränderungen der Emotion ausschlaggebend.

**Gegner steigt aus (fold)** Wenn ein Gegenspieler aus dem Spiel aussteigt, dann ändern sich die Emotionen *Wut*, *Freude* sowie *Angst*. Auch hier wird zunächst nach der Spielphase unterschieden. Da das Ziel eines jeden Pokerspielers ist, seinen Gewinn zu maximieren, ist an dieser Stelle die Unterscheidung nach Kartenwerten sehr wichtig. Ein Spieler würde sich zwar generell freuen, wenn ein anderer Spieler aus einer Runde freiwillig aussteigt, da somit seine Chancen auf einen Sieg erhöht werden, trotzdem ist es jedoch auch so, dass bei eigenen guten Karten der Pot durch den Spieler, der aussteigt, nicht mehr erhöht werden kann. Somit bleibt die Gewinnsumme, die durch den ausgestiegenen Spieler eingebracht wurde, konstant, weshalb die Emotionen des Spielers negativ beeinflusst werden können.

***Wut*** Die Emotion *Wut* verringert sich generell dann, wenn ein Gegenspieler aussteigt und der Spieler selbst eher schlechte Karten hat. Dazu kann natürlich die Charaktereigenschaft *Gegner einschätzen* sowohl positiv als auch negativ auf die Wut wirken.

***Freude*** Auch die Emotion *Freude* hängt von der Charaktereigenschaft *Gegner einschätzen* ab. Auch hier gibt es eine Anpassung in beide Richtungen, abhängig vom Wert für *Gegner einschätzen*.

***Angst*** Die Emotion *Angst* hängt nur von der Phase des Spiels und den Kartenwerten ab. Je weiter fortgeschritten die Spielphase ist und je besser die Karten sind, je größer wird die Angst.

**Gegner hält den Einsatz (check)** Hält ein Gegner den Einsatz, so ändern sich je nach Spielphase und Kartenwert die Emotionen *Angst*, *Wut* und *Freude* nur sehr gering, da es sich um den ersten Einsatz handelt. Charaktereigenschaften der Spieler sind für die Emotionsveränderung bei dieser Gegneraktion nicht relevant. Bei einem schlechten Kartenwert verändert sich nur die Emotion *Angst*. Ist der Kartenwert neutral, so verändern sich *Wut* und *Freude*. Bei guten Karten verändern sich alle drei Emotionsgruppen: *Angst* und *Wut* sinken und die *Freude* steigt.

**Gegner geht mit (call)** Obwohl die Aktion *mitgehen* der Aktion *halten* sehr ähnlich ist, ändern sich durch diese Aktion die Emotionen *Angst*, *Wut* und *Freude* deutlich stärker und hängen auch von den Charaktereigenschaften *Gedächtnis* und *Gegner einschätzen* ab. Natürlich ist auch diese Aktion sowohl von der Spielphase als auch vom Wert der eigenen Karten abhängig.

***Wut*** Die Emotion *Wut* erhöht sich nur bei eher schlechten Karten. Bei guten Karten beruhigt sich der Spieler wieder.

***Freude*** Die Emotion *Freude* hängt stark von der Charaktereigenschaft *Gegner einschätzen* ab, denn wenn der Spieler seinen Gegner gut einschätzen kann, dann weiß er auch, ob der Gegner mitgeht, weil er gute Karten hat, oder weil er bluffen will. Auch hilft hier die Charaktereigenschaft *Gedächtnis* weiter, da der Spieler sich mit einem guten Gedächtnis merken kann, wie die Aktionen des Gegners in der Vergangenheit waren. Sind die Karten des Spielers dann noch gut, so steigert sich seine Freude.

***Angst*** Die Emotion *Angst* ändert sich in jeder Phase. In den frühen Phasen verringert sich die Angst nur bei guten Karten. Je später die Phase ist, je eher erhöht sich die *Angst*. Meistens bei eher besseren Karten. Die Charaktereigenschaft *Gegner einschätzen* hilft hier natürlich weiter, da sie die Angst reguliert.

**Gegner geht All-In** Die Aktion des Gegners *All-In* zu gehen verändert die Emotionen *Wut*, *Freude* und *Angst* sehr stark. Auch hier gilt die Regel: Je später die Spielphase, je mehr verändern sich die Emotionen. Die Charaktereigenschaften *Gegner einschätzen*, *Spielverhalten Hände*, *Gedächtnis* sowie *Setzverhalten Geld* beeinflussen die Emotionen bei dieser Aktion.

***Wut*** Die Emotion *Wut* wird nicht beeinflusst, wenn der Spieler schlechte Karten hat und sich in einer frühen Spielphase befindet. Ein Bluff des Spieler ist nach dieser Aktion ausgeschlossen, es sei denn, er ist der Meinung, dass der Gegner mit noch schlechteren Karten geblufft hat. Die Wut erhöht sich in späteren Spielphasen, da der Spieler zu dem Zeitpunkt schon mehr Chips bezahlt hat und vermutet, dass er mit seinen schlechten Karten nur sehr geringe Siegchancen hat.

***Freude*** Die Emotion *Freude* wird beeinflusst durch die Charaktereigenschaften *Gedächtnis*, *Gegner einschätzen* sowie *Spielverhalten Hände*. Die *Freude* erhöht sich genau dann, wenn der Spieler in einer späten Spielphase gute Karten hat.

***Angst*** Die Emotion *Angst* wird beeinflusst durch die Charaktereigenschaften *Gedächtnis* und *Spielverhalten Geld*. Auch die Angst erhöht sich in späten Spielphasen mit guten Karten ähnlich wie die Freude, denn auch mit guten Karten kann man verlieren.

#### Emotionsveränderungen bei eigenen Aktionen

**Folden** Grundsätzlich werden in diesem Fall nur die Emotionen für *Wut* und *Trauer/Freude* verändert. Dabei spielen die mathematischen Gewinnwahrscheinlichkeiten, welche von PokerTH berechnet werden, und der Betrag, den man seit Erhalt der aktuellen Handkarten gesetzt hat, eine wichtige Rolle. Der Grundgedanke ist, dass man bei schlechten Karten sich eher freut, wenn man nicht viel Geld gesetzt hat, da dieses beim Weiterspielen verloren gegangen wäre. Hat man jedoch viel eingesetzt ist man traurig. *Wut* wird in diesem Fall unverändert gelassen, da in dieser Situation eher die *Trauer* Kernemotion ist, weil man jenem Geld nachtrauert. Wenn die Gewinnwahrscheinlichkeiten hoch sind, hat man mit der Entscheidung zu *folden* potenziell viel aufgegeben. Abhängig davon wie hoch der Betrag ist, den man mit seinen aktuellen Handkarten in das Spiel investiert hat, steigt die *Wut*, da man sich über seine eigene Entscheidung aufregt aus dem Spiel auszusteigen, obwohl die Chancen gut standen, zu gewinnen. Liegt die Gewinnwahrscheinlichkeit ungefähr in der Mitte wird nur der Wert für *Trauer/Glück* in Abhängigkeit des bisher gesetzten Geldbetrages angepasst, da diese starke „Fehleinschätzung", durch den man potenziellen Gewinn trotz guter Chancen verspielt, nicht gegeben ist. Daher trauert man wie in der Situation der schlechten Karten proportional so stark, wie hoch der gesetzte Betrag war. In all diesen Fällen wird ein Berechnungsmodell verwendet, welches eine lineare Funktion beinhaltet. Es wird eine untere und eine obere Schranke für den *Wut*- bzw. *Trauer/Glück*-wert gesetzt. Dieser Bereich stellt die maximale Veränderung jener Emotion dar. Zudem wird eine untere und obere Geld-Schranke festgelegt. Außerhalb dieses Bereichs richten sich die berechneten Werte entsprechend an der oberen, oder unteren Emotionsschranke. Innerhalb des Bereichs wächst/fällt der berechnete Wert linear.

**Checken** In dieser Situation wird nur eine Anpassung an der Emotion *Selbstsicherheit/Nervosität* vorgenommen, da die bewusste Entscheidung weiterzuspielen weniger Einfluss auf *Wut* oder *Freude* hat, sofern kein weiteres Ereignis eintritt.
Es werden wieder die objektiven mathematischen Gewinnwahrscheinlichkeiten für die Berechnung der Emotion verwendet. Dabei fließen schlechtere Wahrscheinlichkeiten weniger stark ein, da man sich bewusst für die Aktion weiterzuspielen entschieden hat.

Die Gewinnwahrscheinlichkeiten werden so transformiert, dass man sie zu dem aktuellen Emotionswert addieren kann. Je besser die Karten sind, desto selbstsicherer wird man. Für schlechtere Karten gilt dementsprechend das Gegenteil, da die niedrigen Gewinnwahrscheinlichkeiten auf einem negativen Zahlenwert abgebildet werden.
Um den Charakter des Spielers in die Berechnung einfließen zu lassen, wird die Charaktereigenschaft *Karten bewerten* als Kerngröße für eine normalverteilte Zufallsvariable verwendet. In diesem Fall hat die Charaktereigenschaft direkten Einfluss auf die Standardabweichung der Zufallsvariable. Dabei gilt, dass je stärker diese Eigenschaft ausgeprägt ist, desto weniger wird der Wert für die Emotionsanpassung verfälscht und umgekehrt. Das bedeutet, dass der Spieler nervöser wird, wenn er schlechte Karten hat und sich bezüglich Kartenverbesserung auch besser einschätzen kann. Zusätzlich wird der Wert noch verändert in Abhängigkeit zur Spielphase, in der sich der Bot befindet. Je näher sich dieser dem *Showdown* nähert, desto nervöser kann er werden.

**Callen/Raisen** Für diese beiden Sitationen wurde das selbe Grundgerüst genommen. Da es eine kritische Situation darstellt, in der man wissentlich Geld riskiert, wird hier nur die Emotion *Selbstsicherheit/Nervosität* verändert. Der Ablauf ähnelt dem Vorgehen wie in der Situation *Checken*, da auch hier die Gewinnwahrscheinlichkeiten als Basis für den Betrag dient, inwieweit sich die Emotion *Selbstsicherheit/Nervosität* verändert. Mit der charakterabhängigen normalverteilten Zufallsvariable wird hier auch der Wert verfälscht, jedoch fließt bei der schlussendlichen Gewichtung nicht nur die aktuelle Spielphase, sondern auch der Betrag ein, den man setzen muss. Abhängig davon ob man nun einen *Call* oder einen *Raise* durchführt wird der Emotionswert anders gewichtet, da man beim *Raise* mehr Eigenverantwortung hat und eine grössere Hemmschwelle überwinden musste um sich zum aktiven Erhöhen des Geldeinsatzes zu entscheiden, als bei der eher passiven Entscheidung „mitzugehen“.

**All-In** In dieser Situation ist es abhängig, wie der Spieler dazu kam auf *All-In* zu gehen. Dies kann auf zwei verschiedene Arten geschehen sein:

1. Der Bot wird durch einen vorher gesetzten Geldbetrag dazu gezwungen *All-In* zu gehen um im Spiel zu bleiben.
2. Der Bot entscheidet sich aus „freien Stücken“ für das *All-In*.

Im ersten Fall wird eine Anpassung an allen Werten vollzogen, da es eine der kritischsten Situationen im Spiel darstellt. Daher ist eine Modifikation der Emotion *Nervosität* gerechtfertigt. Es ist ebenfalls *Freude* zu verändern, da man sich sehr siegessicher sein kann und sich auf den Gewinn freut. Schließlich wird noch *Wut* verändert, da man durch äußere Einflüsse dazu gezwungen wurde sein ganzes Geld einzusetzen, um im Spiel zu bleiben.
Der zweite Fall verhält sich analog, nur dass *Wut* nicht angepasst wird, da man sich bewusst für das *All-In* entschieden hat. In alle Berechnungen fließen, wie zuvor auch, die Wahrscheinlichkeit, das Spiel mit den aktuellen Handkarten zu gewinnen ein, welche mit einer durch die Charaktereigenschaft *Karten bewerten* gesteuerten normalverteilten Zufallsvariablen verfälscht wird.
Die Emotion *Nervosität* wird in beiden Fällen auf die gleiche Weise - jedoch unterschiedlich stark - verändert. Es wird der Quotient aus dem Maximum der Geldbeträge, die alle anderen Spieler noch nicht gesetzt haben, und dem Betrag, den man selbst gesetzt hat mit dem bisherigen berechneten Wert der modifizierten Gewinnwahrscheinlichkeiten multipliziert. Dies hat zur Folge, dass sich die Emotion verstärkt (oder schwächt, wenn der Kehrwert genommen wird), wenn andere Mitspieler mehr Geld besitzen als man selbst gesetzt hat.
Somit wurde modelliert, dass der Bot nervöser, wütender und trauriger wird, je mehr

Geld die Gegenspieler haben im Vergleich zum eigenen Einsatz, denn diese Spieler werden nach einem Verlust ihres Setzbetrages immer noch im Spiel sein. Für den Fall, dass man eigenständig *All-In* geht wird die potenzielle Freude, die man erreichen kann, geschmälert, je höher das Maximum der Geldbeträge, die die Gegenspieler noch besitzen, den eigenen Setzbetrag überbietet.

#### Emotionsveränderungen bei allen anderen Aktionen

##### Handkarten erhalten

Zuerst werden die objektiven und mathematischen Wahrscheinlichkeiten, mit diesen Handkarten zu gewinnen, berechnet. Diese objektive Einschätzung der Karten wird dann mittels dem Charakterwert *Karten bewerten* verfälscht. Die Richtung dieser Verfälschung, ob sie besser oder schlechter eingeschätzt werden, ist zufällig und beide Werte individuell. Je höher der Charakterwert ist, desto weniger wird der objektive Wert verändert. Diese subjektive Einschätzung wird dann mit dem Wert *Spielverhalten der Hände* verrechnet, so dass der Einschätzungwert erneut verringert oder erhöht wird und anschließend die Zufriedenheit mit dem Blatt darstellt. Der Charakterwert codiert, welche Hände gespielt werden und somit auch, mit welchen Hände sich der Spieler zufrieden gibt. Abhängig von dieser Zufriedenheit wird die Emotion *Freude* um maximal vier erhöht oder verringert.
Zusätzlich wird der Durchschnitt der Zufriedenheitswerte über die letzten gemerkten Spiele berechnet. Wieviele Spiele sich der Spieler merken kann, hängt von dem Charakterwert *Gedächtnis* ab. Abhängig vom Durchschnitt wird dann die Emotion *Freude* erneut um maximal vier erhöht oder verringert.

##### Gemeinschaftskarten erhalten

Grundlage bilden wieder die objektiven und mathematischen Gewinnwahrscheinlichkeiten der Karten. Diese werden einmal ohne die neu erhaltenen Karten und einmal mit diesen berechnet, um sie dann, wie beim Erhalt der Handkarten, durch den Wert *Karten bewerten* zu verfälschen. Von diesen beiden subjektiven Einschätzungen wird die Differenz gebildet. Zusätzlich wird mittels dem Charakterwert *Spielverhalten der Hände* bestimmt, ab welchem Wert für die subjektive Einschätzung der Karten sich der Spieler mit ihnen zufrieden gibt.
Wenn sich die Gewinnwahrscheinlichkeit nach der subjektiven Einschätzung verbessert hat, dann kommt es darauf an, ob der Spieler gefoldet hat oder noch im Spiel ist. Ist er noch im Spiel, dann erhöht sich der Wert für *Freude* und, wenn er trotzdem nicht mit den Karten zufrieden ist, der Wert für *Nervosität*.
Hat sich die subjektive Gewinnwahrscheinlichkeit verschlechtert und der Spieler schon gefoldet, so hat er eine richtige Entscheidung getroffen und die *Freude* steigt während die *Nervosität* sinkt. Hat er aber bei einer Verschlechterung nicht gefoldet, dann verringert sich die *Freude* und die *Nervosität* steigt.

##### Nachrichten erhalten

Auch beim Erhalten von Nachrichten wird zuerst, wie beim Erhalten von neuen Karten, eine subjektive Einschätzung der Karten berechnet. Anschließend werden abhängig davon, zu welcher Gruppe die empfangene Nachricht gehört, die Emotionen verändert. Bei gerichteten Nachrichten führt sie nur zu einer Veränderung, wenn der Spieler auch der Empfänger ist. Die existierenden Gruppen verändern die Emotionen wie folgt:

**Freude gerichtet** erhöht die *Nervosität* abhängig von der Bewertung der Karten. Bei der speziellen Nachricht *"Du spielst wie ein Anfänger"* steigt zusätzlich die *Wut* auf den Sender und, wenn der Empfänger außerdem im Besitz von sehr guten Karten ist, auch die *Freude*.

**Freude ungerichtet** führt, wenn der Spieler noch nicht gefoldet hat, zu einer gesteigerten *Nervosität*.

**Schadenfreude gerichtet** verringert die *Freude* und steigert die *Wut* auf den Sender.

**Wut gerichtet** erhöht die *Wut* auf den Sender stark und führt, wenn der Spieler noch im Spiel ist und akzeptable Karten hat, zu einer Verringerung der *Nervosität*.

**Wut ungerichtet** verringert die *Nervosität*, wenn der Spieler noch im Spiel ist und keine schlechten Karten hat.

**Nervosität gerichtet** führt zu einer verringerten *Wut* auf den Sender. Befindet sich der Empfänger noch im Spiel, sinkt seine *Nervosität*, ansonsten steigt seine *Freude*.

**Nervosität ungerichtet** verringert die *Nervosität*, wenn der Empfänger noch im Spiel ist. Bei der speziellen Nachricht *"Was mach ich eigentlich hier?"* wird zusätzlich die *Wut* auf den Sender verringert und, wenn der Empfänger noch mit sehr guten Karten am Spiel teilnimmt, die *Freude* erhöht.

### 3.5.5 Emotionseingabe des Menschen und Visualisierung der Emotionen

Ein weiteres Ziel der Projektgruppe 529 war es, einen neuen Kommunikationskanal zu konstruieren. Über diesen sollten insbesondere die Emotionen signalisiert und dargestellt werden können.

Die erste Idee war eine Kombination aus Audio- und Videosignalen zu nutzen. Dabei sollten die menschlichen Spieler in ein Mikrofon sprechen können und beim Spielen mit einer Kamera, z.B. einer Webcam, aufgenommen werden. Ebenso sollten die Computerspieler Audionachrichten versenden können und ein Video von ihnen anzeigen. Für das Video wären die Projektgruppenmitglieder beim Pokerspielen aufgenommen worden. Davon wären viele kleine Videos erstellt worden, so dass jedes eine andere Situation und andere Emotionen darstellt. Diese Videos wären dann in einer Endlosschleife angezeigt worden, wobei immer das Video gezeigt wird, dass zur aktuellen Spielsituation und den aktuellen Emotionen passt. Diese Idee wurde jedoch schnell wieder verworfen, weil die Interpretation von natürlicher Sprache als Audiosignal und von Videos sehr schwer und nicht Kernaufgabe der Projektgruppe ist.

Eine weitere Idee war es, dass man, ähnlich wie bei *The Palace* (siehe Kapitel A.2.5), einen Avatar besitzt, den man jederzeit ändern kann, wobei man die Auswahl aus einer beliebig großen Sammlung von vordefinierten und selbst erstellten Avataren hat. Wie John Suler (siehe Kapitel A.2.5) festegestellt hat, steht jeder Avatar für Emotionen und/oder für Charaktereigenschaften. Doch auch hier ist wieder das Problem, dass die automatische Interpretation der Avatare sehr schwer ist. Das lässt sich umgehen, indem man nur vordefinierte Avatare zulässt, denen schon eine feste Bedeutung zugewiesen wurde. Genau dieses wurde in Form von Emoticons umgesetzt (siehe Kapitel 3.8.1). Emoticons sind sehr einfache Bilder, die jeder auf Anhieb versteht und die eine sehr große Ausdruckskraft besitzen. Es gab auch Überlegung, sowohl die Emoticons, wie sie implementiert wurden, zusammen mit individuellen Avataren zu benutzen. Die Avatare wären dann allerdings nur ein optischer Zusatz gewesen, weil die Interpretation zu schwer ist. Dieses Modell wurde nicht umgesetzt, weil die in *PokerTH* existierende GUI keinen Platz für zusätzliche Avatare geboten hat und der Aufwand, die GUI für diesen Zweck zu verändern, zu groß gewesen wäre.

An dieser Umsetzung mit den Emoticons gab es auch Kritik. Beim Pokerspielen gegen Menschen ist es eine regelrechte Kunst, die Gesichtsmimik zu kontrollieren. Das wird besonders beim Bluffen benötigt. Durch die Auswahl von Emoticons ist dies jetzt sehr einfach. Das führte zu den Überlegungen, dass der Mensch eine Eingabe machen müsste,

die er nicht so einfach kontrollieren konnte und die Aufschluss über seinen emotionalen Zustand gibt. Auf den ersten Blick schienen biometrische Signale, insbesondere der Puls, dafür geeignet. Doch auch die Interpretation eines Pulssignals ist nicht einfach. Dazu kommt noch, dass wenn man nicht nur die Emoticons anhand des Pulsfrequenzverlaufes automatisches setzen lässt, sondern diesen auch graphisch darstellt, dieser für einen Computerspieler schwer zu generieren ist. Zumindest war die Vermutung sehr stark, dass man einen Computerspieler schnell anhand seiner Pulsfrequenz würde erkennen können. Neben diesen Problemen bei der Implementierung, trat das Problem auf, dass das Pulssignal in den Computer eingespeist werden muss. Pulsmessuhren sind im Sportbereich weit verbreitet. Bei diesen werden die Daten allerdings erst gespeichert und später, nach dem Training, ausgewertet. Für das Pokerprogramm musste die Verarbeitung zeitgleich mit der Aufnahme möglich sein. Nachfragen bei diversen Herstellern dieser Pulsuhren ergaben, dass dies mit den Uhren entweder nicht möglich ist oder die benötigten technischen Informationen nicht veröffentlicht werden dürfen. Auch in der Medizin werden biometrische Daten erfasst. Es existieren mittlerweile diverse EKG-Geräte, z.B. das *BT 3/6* und das *BT 12*[7] der Firma *Corscience GmbH & Co. KG*[8], die man an einen handelsüblichen PC anschließen kann. Die Firma *Corscience GmbH & Co. KG* hätte auch das proprietäre Übertragungsprotokoll mitgeliefert, damit man die verschiedenen Signale, u.A. Puls- und Herzfrequenzrate, mit anderen Programmen verarbeiten kann. Anschaffungskosten von ca. 1000€ überstiegen allerdings das Budget der Projektgruppe deutlich. Zudem hätte eine solche Variante bedeutet, dass jeder, der das erweiterte PokerTH spielen wollte, im Besitzt eines solchen EKG-Gerätes sein müsste. Das ist im privaten Bereich natürlich ausgeschlossen.

Weitere Ideen, um zu überprüfen, ob der Spieler überrascht oder im Stress ist, waren, dass er während des Spieles einen zufällig auf dem Bildschirm auftauchenden roten Punkt mit der Maus anklicken oder in einer bestimmten Frequenz eine Taste drücken muss. Bei Zügen muss der Lokführer alle paar Sekunden mit dem Fuß einen Schalter betätigen, damit sichergestellt ist, dass er sich konzentriert. Beide Ideen wurden nicht umgesetzt, weil sie zu sehr vom eigentlichen Spiel ablenken würden.

Zusätzlich zu den auswählbaren Emoticons sollten auch Textnachrichten verschickt werden. Aufgrund der Schwierigkeit des Turing-Tests (vgl. [Tur50]) war sofort klar, dass keine natürliche Sprache benutzt werden sollte. In der fertigen Implementierung ist es auch möglich, vorgefertigte Nachrichten zu versenden und zu erhalten (siehe Kapitel 3.8.1). Damit die Nachrichten von einen Computerspieler so wirken, als ob sie von einem Menschen geschrieben wurden, sollten das Anzeigen ursprünglich noch besonders gestaltet werden. Zum einen sollte die Tippgeschwindigkeit visualisiert werden. Das bedeutet, dass die Nachricht buchstabenweise eingeblendet wird, so als ob sie zeitgleich getippt wird. Zum anderen sollte sie mit künstlichen Rechtschreibfehlern versehen werden. Für einen Computerspieler wären dazu zwei weitere Charaktereigenschaften, eine für die Tippgeschwindigkeit und eine für die Rechtschreibung, nötig. Für Nachrichten von Menschen hätten diese Werte vor dem Spiel anhand eines kurzen Textes, den der Spieler abtippen muss, bestimmt werden können. Um den Spieler nicht zu irritieren, dass er eine fehlerfreie Nachricht aus dem Menü auswählt, die anschließend mit Rechtschreibfehlern im Nachrichtenfester angezeigt wird, sollten eigenen Nachrichten fehlerfrei angezeigt werden. Nur die anderen Spieler würden eine fehlerbehaftete Nachricht sehen. Aus Zeitgründen wurden diese optischen Eigenschaften nicht umgesetzt.

Für die Verbindung von Emoticons und Nachrichten gab es drei Modelle:

1. Emoticons und Nachrichten können unabhängig von einander gewählt werden.
2. Zu einem gewählten Emoticon wird automatisch einen passende Nachricht ge-

[7] `http://www.corscience.de/de/medizintechnik/produkte-systeme/herz-kreislauf-medizin/bt-pc-ekg.html` [22.07.08]

[8] `www.corscience.de`

schickt.

3. Zu einer geschickten Nachricht wird automatisch ein passendes Emoticon angezeigt.

Mit passend ist gemeint, dass beides für die selbe Emotionsgruppe steht. Bei einer Abstimmung wurde das erste Modell gewählt. In der fertigen Implementierung wurde es allerdings wieder etwas eingeschränkt, so dass man erst ein Emoticon auswählen muss, um danach Nachrichten versenden zu können, die der durch das Emoticon signalisierten Emotion entsprechen (siehe Kapitel 3.8.1).

## 3.6 KI beim Poker: Einleitung

Naive Ansätze beim Entwurf von emotionalen Computergegner für Poker bestehen darin, einfache Charaktereigenschaften eines Mitspielers zu repräsentieren. Hierbei wird versucht Charaktereigenschaften wie z.B. vorsichtig oder agressiv zu simulieren, wobei ein vorsichtiger Spieler eher bereit ist ein Blatt aufzugeben als ein agressiver Spieler.

Außerdem basieren viele Entscheidungen konventioneller Pokerbots überwiegend auf der statistischen Analyse der einsehbaren Blätter. So besteht der größte Teil der Entscheidungsroutinen des PokerTH-Bots in der Überprüfung der Handstärke und des Handpotentials zur möglichen Erweiterung zur Straße oder einem Flush. Beim Testspiel gegen den PokerTH-Bot fiel den Probanden auf, dass sich die Spielweise nicht signifikant veränderte und es so möglich war die Spielweise des Bots leicht zu durchschauen. Beispielsweise fiel schnell auf, dass der PokerTH-Bot mit einer guten Hand agressiv spielte und mit einer wenigen guten Hand sehr defensiv. Der Spieler wusste demzufolge sehr genau wie die Hand des Bots bei beobachteter Spielweise wahrscheinlich auszusehen hatte.

Im Gegensatz zu den rein statistischen Methoden des PokerTH-Bots war es unser Ziel mit Hilfe von CI-Methoden einen dynamischeren Computergegner zu entwerfen. Am Anfang des Projektes wurde ein Blackboxmodell entworfen, welches die Schnittstelle zum Gesamtsystem vorbereiten sollte. Innerhalb dieser Blackbox wurde ein Vier-Phasen-Modell angestrebt, so dass sich die KI aus vier kleineren KIs zusammensetzen sollte. Jede dieser KI sollte sich um die Entscheidungsfindung in einer der Spielphasen Preflop, Flop, Turn, River bemühen. Die Begründung dieser Aufteilung lag darin, dass beim Pokerspiel zu den verschiedenen Zeitpunkten je nach Anzahl der offen gelegten Karten unterschiedlich viele Informationen zur Verfügung stehen und man hoffte damit sowohl die Komplexität des Gesamtsystems durch Einsatz mehrere Teilsysteme reduzieren zu können, als auch den in den unterschiedlichen Spielphasen verschiedenen notwendigen Strategien gerechter zu werden.

Die Blackbox sollte als Input sämtliche Tischinformationen über die einzelnen Spieler wie Anzahl der Spielchips und gegenwärtigen Einsatz erhalten und zudem auch Einschätzungen wie zum Beispiel die Bluff-Wahrscheinlichkeit. Die Tischinformationen sollten durch eine zentrale Komponente geliefert werden, welche die Spielinformationen weiterleitete und die Schnittstelle zwischen unserem Programm und der PokerTH-Umgebung war.

Da zu diesem Zeitpunkt noch nicht klar war, welche CI-Methode(n) für das Pokerspiel am besten geeignet wären, probierte die Gruppe zunächst aus, ein Neuronales Netz zu benutzen. Bei diesem handelte es sich um ein Feedforward-Netz bestehend aus einer Eingabeschicht, einer versteckten Schicht und einer Ausgabeschicht mit welchem man hoffte komplexere Zusammenhänge erlernen zu können. Um das Netz in der Entscheidungsfindung nicht einzuengen, entschied man sich beim Netz zunächst sämtliche Kanten zwischen den Schichten zuzulassen und bei Bedarf später die Netztopologie noch zu ändern

bzw. zu erweitern. Zum Lernen wurde ein rudimentärer Backpropagation-Algorithmus genutzt welcher die Kantengewiche dahingehend anpasste, dass in der Lernphase der Ausgabevektor des Netzes einem vorgegebenen Zielvektor zunehmend angeglichen wurde. Zu diesem Zeitpunkt umfasste der Ausgabevektor Werte für *raise*, *call*, *check*, *bet* und *fold* sowie Werte um die geplante Kommunikation in Form einer Textausgabe zu realisieren. Hierfür umfasste das Netz für jede Emotionsklasse einen Wert, welcher die Ausprägung der Emotion widerspiegelte. Die Emotion mit dem größten Wert wurde unter Berücksichtigung anderer Charaktereigenschaften wie Kommunikativität dann zu einer Textausgabe verarbeitet, wobei es natürlich auch möglich war, dass kein Text ausgegeben wurde.

Da es zu diesem Zeitpunkt große Probleme gab, wie die Zielfunktion lautet, welche es zu optimieren galt, wurde ein weiterer Ansatz entwickelt. Die Idee kann wie folgt beschrieben werden:

- Es wird eine History angefertigt, welche die Spielsituationen mitlogt.
- Auf Basis dieser Datensätze wird offline ein Netz antrainiert.
- Hierbei gilt, dass das Verhalten in einer Situation gut war, wenn unter gegebenen Inputfaktoren das Netz die gleiche Handlung wie die mitgelogte Handlung ausführt. Es geht bei der Optimierung des Netzes nicht darum möglichst viel Geld zu verdienen, sondern ein dem Menschen möglichst ähnliches Verhalten zu erlernen.
- Neben dem noch zu erweiternden Backpropagation-Algorithmus wurde ein weiterer konkurrierender Ansatz ausprobiert. In diesem sollten über einen evolutionären Algorithmus neuronale Netze mit unterschiedlichen Kantengewichten generiert werden, um so die Kantengewichte zu optimieren.

Beim Testen der beiden Methoden stellte sich heraus, dass der evolutionäre Algorithmus wesentlich bessere Ergebnisse als der Backpropagation-Algorithmus lieferte, weshalb dieser Ansatz am Ende erfolgreich zum antrainieren unserer Computergegner eingesetzt wurde. Da die Textausgabe mit Hilfe der Entscheidungsbasis des Netzes nicht die gewünschte Qualität erzielte entschied man sich diese separat durch ein Regelsystem zu kontrollieren.

### 3.6.1 Evolutionärer Algorithmus

Der bisher verwendete Lernalgorithmus lernte online, d.h. er passte immer dann das Neuronale Netz an, wenn es eine Fehlklassifikation beging. Die verwendete Backpropagationmethode lernte dadurch sehr langsam und qualitativ auch nicht besonders gut, da sich die Annäherung an relativ leichte Verhaltensmuster als schwierig herausstellte. Daher entschied man sich, einen Evolutionären Algorithmus zum Lernen der Neuronalen Netze zu verwenden. Die Neuronalen Netze stellen die Individuen des *EA* dar. Der Lernalgorithmus wurde in die Klasse der Neuronalen Netze integriert. Um Individuen für das Lernverfahren und die Netze, welche später als Entscheidungsinstanzen verwendet wurden, zu trennen, wurde eine neue Klasse angelegt, welche die Netze geeignet repräsentiert. (vgl. Abbildung 6)

Der EA wurde mit Steuerparametern implementiert, um die Mächtigkeit des Lernverfahrens beliebig skalieren zu können, wobei die „Plus-Selektion“ fest integriert wurde. Folgende Parameter konnten übergeben werden:

**Elternindividuen** $\mu$: Die Anzahl der Elternindividuen. Je größer sie ist, desto größer wird die Population.

**Kinderindividuen** $\lambda$: Die Anzahl der Kinderindividuen. Je größer $\lambda$ wird, desto mehr Kinder werden pro Generation gebildet und desto größer ist damit auch die Ge-

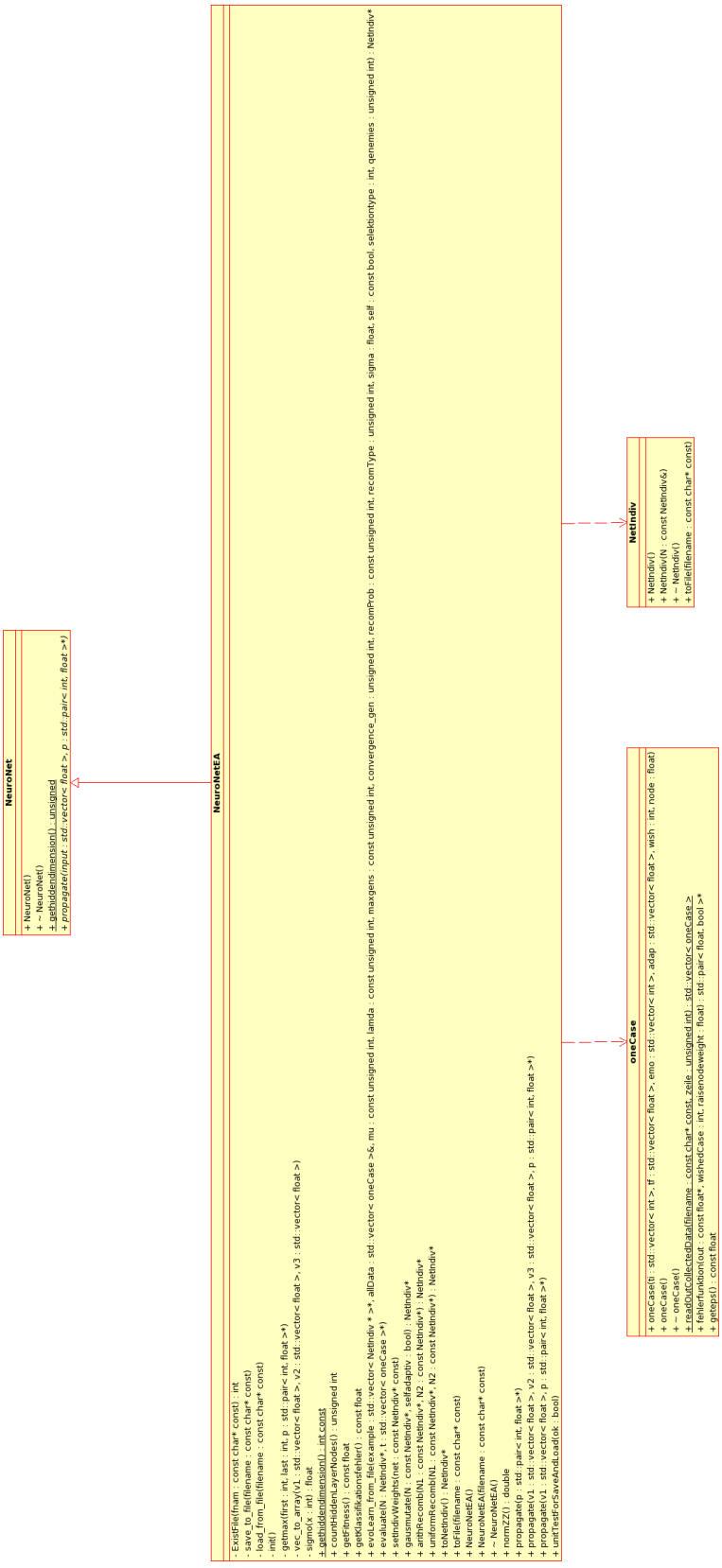



**Abbildung 6:** Klassendiagramm des Evolutionären Algorithmus

samtpopulation. Problematisch wird es, wenn die Auswertung eines Individuums einen hohen Rechenaufwand braucht. So werden bei geringerem $\mu$ und hohem $\lambda$ viele Individuen gebildet, ausgewertet und wieder gelöscht.

**Generationszahl:** Die Anzahl an Generationen, die der EA maximal rechnen soll.

**Generationsanzahl für Konvergenz:** Die Anzahl an Generationen, in denen neue *und* bessere Individuen „entdeckt" werden müssen.

**Rekombinationstyp:** Die Auswahl zwischen *arithmetische* und *uniforme Rekombination*. *Artihmetische Rekombination* verwendet ein Vektormodell und bildet zwei Elternindividuen in einen Raum ab. Das Kindindividuum wird bei dieser Rekombination so gewählt, dass es in dem Vektorraum auf der Strecke zwischen den beiden Elternindividuen zu finden ist. An welcher Stelle auf der Strecke genau es sich befindet, wird zufällig gewählt. Bei der *uniformen Rekombination* wird für jeden „Baustein" des Kindindividuums zufällig ausgewählt, ob das eine Elternindividuum als Vorbild fungiert oder das andere.

**Rekombinationswahrscheinlichkeit:** Gibt die Wahrscheinlichkeit an, mit der eine Rekombination anstelle einer reinen Mutation durchgeführt wird. Nach jeder Rekombination wird das erzeugte Individuum anschließend nochmal mutiert.

**Schrittweite $\sigma$:** Gibt die „Schrittweite" an, mit der die Mutation erfolgt. Je höher $\sigma$ ist, desto stärker unterscheidet sich das mutierte Kindindividuum von seinem Elter (im Fall der Rekombination: von seinen Eltern).

**Selbstadaption** Gibt an, ob die Standardabweichung $\sigma$ mit Hilfe der Selbstadaption gesteuert wird. Bei der Selbstadaption wird mit jeder neuen Mutation $\sigma$ verändert (mutiert). Die Idee dabei ist, dass sich die Schrittweite guter Individuen durchsetzt und diese Schrittweite genutzt werden kann, um bessere Individuen zu finden.

**Selektionstyp:** Gibt den Selektionstyp an, mit dem die Selektion der Überlebenden durchgeführt wird. Es konnten die Selektionstypen *Bestenselektion* und *q-stufige Turnierselektion* ausgewählt werden. Für letzteres Verfahren konnte man noch die Größe für $q$ angeben.

**Datensatz:** Enthält die Menge der Testdaten in einer Datenstruktur. In Abhängigkeit von der Menge der Testdaten steigt auch der Aufwand für die Auswertung eines *Netz-Individuums*.

**Startindividuen:** Ermöglicht bereits trainierte Individuen in einen neuen evolutionären Optimierungsprozess einzufügen. Dadurch können vorhandene gute Lösungen evtl. noch weiter verbessert werden.

Da die Netzgröße mit 300 Eingabe-, 300 versteckten und 11 Ausgabeneuronen auf $300^2+300\cdot 11 = 93.300$ Kanten anstieg, war früh deutlich, dass der EA ein so hochdimensionales Problem nicht in akzeptabler Zeit lösen konnte. In Absprache mit den anderen AGs wurde die Informationsmenge der Module für die Eingabe stark reduziert. Ebenso wurde bemerkt, dass viele Aktionsmöglichkeiten zusammengefasst werden konnten. So haben *Check* und *Call* ein ähnlich passives und *Raise*, *Bet* und *All-In* ein ähnlich aktives Verhalten. Auf diese Weise konnte das Problem auf eine moderate Variablenmenge von $47^2 + 47 \cdot 3 = 2350$ Kanten reduziert werden.

Mit dem neuen Lernverfahren wurde das Konzept des *offline-learning* angestrebt. Ziel war es, die Netze so zu trainieren, dass sie sich möglichst gut an einen Trainingsdatensatz anpassen konnten. Zu Anfang war eine reine Klassifikiation in drei Fälle *Fold*, *Check/Call* und *Raise/All-In* vorgesehen. Die Ausgabeknoten stellten mit ihren Werten ein Binärmustern wie z.B.$(1;0;0)$ für *Fold* oder $(0;1;0)$ für *Check/Call* dar.

Um aber dem Netz die Möglichkeit zu geben, den Setzbetrag für *Raise* (bzw. die Entscheidung, wann einfacher *Raise* oder ein *All-In* gemacht wird) festzulegen, sollten die Werte, die bei den entsprechenden Knoten im *Neuronalen Netz* gesammelt werden, erhalten bleiben und *nicht* mit einem Schwellwert auf die Menge $\{0; 1\}$ normiert werden. Die Klassifizierung einer bestimmten Situation hing nun von dem maximalen Wert aller Ausgabeknoten ab. Der entsprechende Fall, den dieser Knoten repräsentiert, stellt dann die berechnete Entscheidung des Netzes dar.

Mit Einführung der reellen Werte an den Ausgabeknoten musste auch die Zielfunktion angepasst werden. Einerseits sollten die Netze so optimiert werden, dass der Knotenwert der gewünschten Situation maximal ist, andererseits sollte im Falle das *Raise/All-In* sich der Knotenwert an den geforderten Wert annähern, welcher den Setzbetrag repräsentiert. Die Zielfunktion bestrafte jeden Fall, in dem beide Sitationen nicht erfüllt waren, mit einem quadratischen Fehlerterm proportional zur „Verletzung“ des nicht erfüllten Trainingsfalls.

Um die Qualität des EAs einzuschätzen, wurden die Netze mit Hilfe des EA an statistischen Datensätzen aus dem Internet trainiert. Diese Datensätze beinhalteten EEG-Messungen von Epilepsipatienten und Zelleigenschaften, welche Brustkrebs identifizieren sollten.

Der erste Datensatz beinhaltete pro Beobachtung 9 Attribute, welche einen vierstelligen, positiv ganzzahligen Wertebereich hatten und immer 2 Klassen zugeordnet wurden. Der Datensatz umfasste 265 Beobachtungen mit 100 Beobachtungen, welche die Beobachtung als ein epileptisches Ereignis klassifizierten, und 165, die das nicht taten.

Der zweite Datensatz umfasste 457 Datensätze, die eine binäre Klassifizierung in „gutartige“ und „bösartige“ Krebszelle erlaubten. Ausgehend von 9 Attributen, welche einen ganzzahligen Wertebeich von eins bis zehn zuließen, sollte die Klassifikation erfolgen.

Für beide Datensätze wurde ein faktorielles Testdesign verwendet, um den EA mit mehreren Parameterkombinationen zu testen. Dabei wurde der EA mit unterschiedlichen Werten für die Generationszahl (50; 100), die Rekombinationswahrscheinlichkeit (0,2; 0,8), die Anzahl der Eltern- (5; 15) und der Kinderindividuen (5; 15) gestartet. Die Werte für die Schrittweite ($\sigma = 0,3$), der Selbstadaption („keine“), des Rekombinationstyps („arithmetische“) und des Selektionstyps („Bestenselektion“) blieben dabei konstant.

Bei beiden Datensätzen konnte der EA gute Lernergebnisse erziehlen.

Zusätzlich wurde das Lernverfahren mit statischen Funktionen überprüft. Das Netz sollte bezüglich Eingabe und Ausgabe verschiedene Funktionen lernen (z.B. Polynome, konstante Funktionen oder Relationen zwischen den Ausgabewerten). Dabei stellte sich heraus, dass das Antrainieren von konstanten Funktionen mit sehr guten Ergebnissen realisiert wurde. Es war gefordert, dass eine bestimmte Aktion (z.B. *Fold*) in jeder Spielsituation ausgeführt werden sollte. Hingegen wurde die Gleichheitsrelation zwischen Knoten (als Bot könnte sie das Verhalten „spiele gleichwahrscheinlich *Call/Check* und *Fold*“ auslösen, wenn die beiden Knoten einen gleichen Wert haben und bei Entscheidungen einer der beiden Knoten nichtdeterministisch ausgewählt wird) mit nicht so befriedigenden Ergebnissen gelernt. In diesem Fall war es wichtig, dass sich ein Ausgabewert eines oder mehrere Knoten an einen bestimmten Wert annähern sollte und nicht wie im oben beschriebenen Fall ein Wert größer als alle anderen sein musste. Das Lernen der Polynomfunktionen gestaltete sich ähnlich schwierig, da auch hier wieder eine Annäherung an einen Funktionswert notwendig war.

Ein weiterer Verbesserungsansatz war die Möglichkeit, bereits berechnete Netze wieder in den Evolutionsprozess einzufügen, um sie noch weiter zu optimieren. Es gab jedoch technische Schwierigkeiten, aufgrund derer die Netze nach erstmaliger Berechnung im neuen Evolutionsprozess ganz anders bewertet wurden. Dieser Bug konnte nicht mehr

identifiziert werden und wurde als nicht gravierend eingestuft, da auch ohne diese Möglichkeit akzeptabele Netze trainiert werden konnten.

### 3.6.2 Neuronales Netz

Das im ersten Projekt für die künstliche Intelligenz eingesetzte Neuronale Netz (siehe Grundlagen: Neuronale Netze 2.2 auf Seite 3) ist ein 47-47-3 *Feed-Forward-Netz*. Wie üblich besteht das Netz aus drei Schichten: der Eingabeschicht, bestehend aus 47 Neuronen, der verborgenen Schicht mit ebenfalls 47 Neronen und der Ausgabeschicht aus lediglich drei Neuronen. In unserem Fall wurde ein Netz mit nur einer verborgenen Schicht gewählt, da dies für die gegebene Problemstellung als ausreichend angesehen wurde. Weiterhin sind die Neuronen der einzelnen Schichten vollständig vermascht und die Verbindungen mit reelwertigen Gewichten aus dem Bereich $[-1, 1]$ versehen. Dies wurde in der Implementierung mittels Adjazenzlisten realisiert.

Die Aktivierungsfunktion der Neuronen jedoch weicht von normalerweise gebräuchlichen Modellen ab. So kommt bei den Neuronen der verborgenen Schicht eine Schwellenwertfunktion mit dem Schwellenwert 0.5 und dem konstanten Ausgabewert 1 zum Einsatz, nicht jedoch bei den Neuronen der Ausgabeschicht. In den drei Neuronen dieser Schicht werden die Ausgaben der Hidden-Layer-Neuronen lediglich aufsummiert und danach separat weitergehend ausgewertet. Bei einem Ausgabewert von 0 oder 1 der 47 Neuronen aus der verborgenen Schicht und Kantengewichten aus dem Intervall $[-1, 1]$ ergibt dies einen möglichen Wertebereich von $-47$ bis 47. Die Neuronen der Eingabeschicht haben, wie bei Feed-Forward-Netzen üblich, die Identität als Ausgabefunktion. Die Eingaben werden also unverändert an die nächste Schicht weitergegeben. Die 47 Eingabewerte für das Netz sind nun wie folgt aufgeteilt (die ersten 14 Werte sind pokerspezifische Werte):

- **Handkarte 1** - die erste eigene Karte
- **Handkarte 2** - die zweite eigene Karte
- **die Flopkarten** - drei an der Zahl, falls vorhanden, sonst $-1$ jeweils
- **Turnkarte** - die Turnkarte, falls vorhanden, sonst $-1$
- **Riverkarte** - die Riverkarte, falls vorhanden, sonst $-1$
- **Spielphase** - *Pre-Flop*, *Flop*, *Turn* oder *River*
- **Button** - der eigene Button, also *Dealer*, *Smallblind*, *Bigblind* oder gar keiner
- **das eigene Geld** - prozentual zum gesamten Geld im Spiel
- **Summe(Einsatz, Hand)** - der bisherige Einsatz in der Runde mit den momentanen Handkarten
- **der Pot** - Summe aller bisherigen Einsätze von allen Spielern, prozentual zum gesamten Geld im Spiel
- **Erhöhungraise** - wieviel Prozent des eigenen Geldes muss mindestens gesetzt werden, um im Spiel zu bleiben,

  (notwendiger Einsatz)/(eigenes Geld)
- **Odds** - die objektive, mathematische Chance zu gewinnen

Diesen Werten folgen die Emotionswerte des Bots:

- **Wut**
- **Trauer/Glück**
- **Selbstsicherheit/Angst**

Den größten Anteil stellen schließlich die vom opponent-modelling ermittelten Werte. Bei ihnen handelt es sich um 30 Stück, die sich jeweils aus fünf Werten für jeden der maximal sechs Gegenspieler zusammensetzen. Ist ein Gegner nicht im Spiel, etwa weil er gefoldet hat, aus dem Spiel ausgeschieden ist oder deshalb, weil mit weniger Spielern gespielt wird, so sind die entsprechenden Werte gleich Null. Die fünf Werte jedes Spielers sind die folgenden:

- **Spielereinschätzung (strategisch)** - Einteilung in loose/tight bzw. aggressive/passive Klassifikationen (siehe Abschnitt 3.7.5 auf Seite 36)
- **Spielereinschätzung (taktisch)** - die Spielereinschätzung aus taktischer Sicht
- **Charakterwert (Ehrlichkeit und Zögerlichkeit)** - 2/3 Ehrlichkeit + 1/3 Zögerlichkeit
- **Charakterwert (Aggressivität und Risikobereitschaft)** - 1/2 Aggresivität + 1/2 Risikobereitschaft
- **Charakterwert (Fähigk. zur Kartenbewertung)** - Produkt aus den beiden diesbezüglichen Werten, siehe Abschnitt Charakterwerte und Parameter 3.7.1 auf Seite 33

Für jeden Spielzug werden diese Werte nun in das neuronale Netz eingespeist. Die resultierenden Werte an den drei Ausgabeknoten werden nun so bewertet: Jeder Knoten steht für eine, von der Spielsituation abhängige, Handlung. Der erste Knoten hat die Bedeutung *fold*, der zweite *call* bzw. *check* und der dritte *bet*, *raise* bzw. *all-in*. Gewählt wird schließlich die Aktion jenes Knotens, an dem der höchste Wert anliegt. Ist bei der Wahl des *raise/all-in*-Knotens ein All-In nicht zwingend, aber der Knotenwert übersteigt eine gewisse vorgegebene Grenze, so wird All-In als Aktion gewählt. Für das Raisen wird der Knotenwert auf den Bereich $[0, 1]$ normiert, danach mit dem Charaktwert „Setzverhalten Geld“ verfälscht (min. 0 und max. 1), und dann als Prozentwert vom halben Geld, das der Spieler besitzt, interpretiert. Dies wird dann gesetzt.

Damit das Netz sinnvolle Ausgaben liefert, muss es trainiert werden. Bei den Trainingsmethoden werden zwei generelle Verfahren unterschieden: Offline- und Online-Training. Beim Online-Training wird das Netz trainiert, während es im Einsatz ist. Diese Methode hat den Vorteil, dass das Netz dynamisch lernen und reagieren kann, erfordert allerdings eine entsprechende Funktion zur dynamischen Bewertung der Aktionen, nach der gelernt werden kann. Außerdem ist dieser Ansatz nur für kleinere Netze und geringe Datenmengen praktikabel, da die entsprechenden Algorithmen sehr rechenintensiv sind. Aus diesen Gründen fiel die Entscheidung für unser Projekt auf ein Offline-Trainingsverfahren. Das Training des neuronalen Netzes, was aus dem Anpassen der Synapsenwerte besteht, wurde zunächst mittels eines Backpropagation Algorithmus durchgeführt. Aufgrund nicht zufriedenstellender Ergebnisse wurde dieser jedoch im Laufe des Projekts durch einen evolutionären Algorithmus (EA) ersetzt (siehe den Grundlagenartikel 2.1 (Seite 2) und den Artikel zum Lernverfahren 3.6.1 (Seite 27)).

Die benötigten Trainingsdaten wurden nun gewonnen, indem die Spielzüge eines realen Spielers in PokerTH in einer Logdatei aufgezeichnet wurden. Dies hat zum einen den Vorteil, dass schnell und einfach eine genügend große Menge Trainingsdaten gesammelt werden konnten, und zum anderen, dass direkt von menschlichem Verhalten gelernt wurde, welches zu simulieren ja erklärtes Ziel der PG war. Außerdem konnten somit einfach verschiedene Spielertypen erstellt werden, wie z. B. ein vorsichtiger, änstlicher oder ein aggressiver Spieler.

Gegen Ende des Projekts wurden die neuronales-Netz-Klassen noch einmal grundlegend erweitert, unter anderem im Hinblick auf Flexibilität und somit auf einen möglichen Einsatz im zweiten Projekt. So wurden sowohl die Anzahl der Neuronen in den Schichten, als auch die Anzahl der verborgenen Schichten zur Laufzeit variabel gemacht. Außerdem

wurde neben der bisher für die Neuronen benutzte Schwellenwertfunktion eine sigmoide Funktion implementiert, die an Stelle dieser eingesetzt werden kann. Im Zuge dessen wurde auch der Backpropagation Algorithmus noch einmal komplett neu implementiert. Diese Änderungen flossen jedoch nicht mehr in das erste Projekt ein, sondern stehen nun für das zweite Projekt zur Verfügung.

## 3.7 Opponent Modeling

Aufgabe der AG3 war es, eine Einschätzung der Gegenspieler zu liefern, sowohl was deren Charakter bzw. Emotionen, als auch was ihre Spielweise und Strategie betrifft. Dies ist auch unter dem Namen „opponent modeling“ bekannt. Basierend auf den Einschätzungen, die wiederum ausschließlich auf den Handlungen der Gegner während des Spiels ermittelt werden, sollte nun für die künstliche Intelligenz (auch „Bot“ genannt) ein Gedächtnis aufgebaut werden, um auf das Verhalten der Gegenspieler sowohl über kurze, als auch über längere Zeiträume reagieren zu können. Eine Funktion zum Speichern der ermittelten Werte in eine Datei sollte dies auch über einzelne Spiele hinaus ermöglichen.

### 3.7.1 Charakterwerte und Parameter

Ganz zu Beginn stellte sich natürlich zunächst einmal die Frage, welche Charakterwerte überhaupt sinnvollerweise modelliert werden sollten. Nach einigen Überlegungen fiel die Entscheidung auf die folgenden Werte:

- *Ehrlichkeit*
- *Risikobereitschaft* (bzgl. Karten)
- *Aggressivität* (bzgl. Geld)
- *Zögerlichkeit*
- *die Fähigkeit, die Gewinnwahrscheinlichkeit der Karten einzuschätzen*
- *die Fähigkeit, das eigene Blatt nach Regeln zu bewerten*

Ehrlichkeit bezieht sich hier vor allem auf das Setzverhalten im Bezug auf die Karten, kurz gesagt, wie hoch die Wahrscheinlichkeit ist, ob ein Spieler nur blufft oder tatsächlich gute Karten auf der Hand hat. Risikobereitschaft soll Auskunft darüber geben, inwiefern ein Spieler bereit ist weiterzuspielen und damit Geld zu riskieren, auch wenn sein eigenes Blatt eventuell keine großen Gewinnchancen verspricht. Die Aggressivität hingegen bezieht sich auf das reine Setzverhalten: erhöht (raise) der Spieler häufig, und setzt er hohe Summen? Die Zögerlichkeit sagt aus, ob ein Spieler schnell handelt oder erst gründlich seine Entscheidungen abwägt. Die letzten beiden Werte schließlich beziehen sich auf die Fähigkeit eines Spielers, Karten zu bewerten. Einerseits wie hoch die Gewinnwahrscheinlichkeit der vorhandenen Karten ist, andererseits wie gut ein Spieler mit den Regeln vertraut ist und somit in der Lage ist, die Karten nach diesen zu beurteilen.

Zusätzlich zu den Charakterwerten sollten aber auch noch Werte modelliert werden, die sich auf das pokerspezifische Verhalten beziehen. Hier fiel die Wahl auf die vier Werte *vpip* (Abkürzung für engl. „voluntarily put money into pot“), *pfr* (percentage of pre-flop raises/pre-flop aggression), *af* (post-flop aggression), und *wtsd* (went till showdown). *Vpip* gibt hierbei Auskunft darüber, ob ein Spieler trotz schlechter Karten weiterspielt und somit Geld in den Pot gibt. Die Werte *af* und *pfr* sagen aus, wie aggressiv die Spielweise eines Spielers ist, also wie häufig er seine Einsätze erhöht und wieviel Geld

er dabei setzt, aufgeteilt in die Pre- und Post-Flop-Phase. *Wtsd* schließlich bezieht sich darauf, wie oft ein Gegenspieler bis zum Ende dabeigeblieben ist. Mit Hilfe dieser vier Werte lassen sich bereits Einteilungen in verschiedene Spielertypen vornehmen. Genaueres zu den Spielertypen sowie den Werten und ihrer Herleitung findet sich im Abschnitt Spielverhalten (siehe 3.7.5 auf Seite 36).

### 3.7.2 Verknüpfung von Charakterwerten und Parametern

Nachdem diese Frage geklärt war, stellte sich automatisch die nächste: Welche Parameter stehen zur Verfügung bzw. welche Parameter sind relevant, um die genannten Charakterwerte zu bestimmen? Die als relevant erachteten Parameter lassen in zwei Kategorien aufteilen: Die spielerbezogenen Werte und allgemeine (rundenbezogene) Werte.

**Spielerbezogene Werte**

**Position** eines Spielers in der Runde

**Wartezeit** wie lange braucht der Spieler bis er handelt, nachdem er dran ist

**Häufigkeit des Raisens**

**Höhe der Beträge** beim Raisen

**Phase des Aussteigens** pre-flop, flop, turn oder river?

**Blatt** die Karten eines Gegenspielers (falls offenbart)

**Cash** das Geld eines Spielers (in % vom Gesamtcash im Spiel)

**Allgemeine rundenbezogene Werte**

**Höhe des Pots** (in %) das Geld, um das in der aktuellen Runde gespielt wird

**Aktuelle Spielphase** Pre-Flop, Flop, Turn oder River

**Gemeinschaftskarten** je nach Spielphase keine, drei, vier oder fünf

**Höhe der Blinds** (in %) die Höhe der Mindesteinsätze

Schließlich mussten die Charakterwerte und die Parameter noch in einen sinnvollen Zusammenhang gebracht werden. Die für einen Charakter relevanten Werte werden bei der Berechnung summiert. Außerdem bekommt jeder Parameter einen Koeffizienten, der es erstens erlaubt, die einzelnen Werte unterschiedlich zu gewichten und zweitens erlaubt, ein proportionales oder eine antiproportionales Verhältnis von Parameter zu Charakterwert darzustellen. Zur besseren Erklärung hier exemplarisch die Berechnung für die Ehrlichkeit:

$$ehrlichkeit = -0.2 \cdot Wartezeit - 1.0 \cdot Phase\ des\ Aussteigens + 1.2 \cdot Cash$$

Die Wartezeit des Spielers sowie die Phase des Aussteigens fließen hierbei negativ ein, d.h. eine längere Wartezeit oder eine späte Phase des Aussteigens sorgen für einen niedrigeren Ehrlichkeitswert. Eine lange Wartezeit könnte nämlich bedeuten, dass der Spieler zögert, weil seine Karten nicht gut sind, und er überlegt trotzdem weiterzuspielen und eventuell zu bluffen. Gleichermaßen wird ein frühes Aussteigen als ehrlich gewertet. Hat ein Spieler andererseits viel Bargeld, so kann er sich eher leisten auch mit mittelmäßigen Karten mitzuspielen, weswegen der Parameter mit positiver Gewichtung hier einfließt. Für jede Charaktereigenschaft wurde außerdem ein 'neutraler' Startwert angenommen, der durch diese Berechnungen erhöht bzw. erniedrigt wird. Die kompletten Beziehungen

zwischen Charakterwerten und Parametern können der Tabelle im Anhang entnommen werden.

### 3.7.3 Grundmodell

Doch nachdem die grundlegenden Werte und die Berechnungsvorschriften standen, mussten weitere Fragen zum Grundmodell geklärt werden. Zunächst stand die Entscheidung an, ob jeder Spieler einzeln modelliert werden sollte oder ein Gesamtwert für den Tisch. Das erste Modell hat den Vorteil, dass die Modellierung genauer ist und zudem Spieler während des Spiels rausfallen. Gleichzeitig ist die zu berechnende, verwaltende und auszuwertende Datenmenge weitaus größer. Für die Modellierung eines Durchschnittswerts über den ganzen Tisch spricht hingegen, dass eben der ganze Tisch für Entscheidungen berücksichtigt werden muss und nicht nur einzelne Spieler. Auch das Datenvolumen ist geringer und somit leichter zu verarbeiten. Die Entscheidung fiel schließlich zugunsten eines hybriden Ansatzes. Jeder Spieler wird einzeln modelliert und aus den einzelnen Werten kann dynamisch eine Bewertung für die Situation am gesamten Tisch berechnet werden. Die nächste Frage war, über welchen Zeitraum während des Spiels die Werte erfasst, ausgewertet und gespeichert werden sollten. Die Wahl fiel auf eine Hierarchie, bestehend aus drei Ebenen für die pokerspezifischen Strategiewerte bzw. zwei Ebenen für die Charakterwerte. Die Strategiewerte wurden aufgeteilt in eine operative, eine taktische und eine strategische Ebene. Die operative Ebene ist für die kurzfristige Bewertung zuständig, die taktische für die mittelfristige und die strategische für eine langfristige Bewertung eines Gegenspielers. Diese Hierarchie, ihre Verhältnisse untereinander und die Berechnung werden im Abschnitt Spielverhalten (siehe 3.7.5 auf Seite 36) genauer erläutert. Die Charakterwerte sind prizipiell identisch unterteilt, jedoch fällt die operative Ebene hier weg. Dies ist darin begründet, dass hier die Zuordnung von Parametern zu den Werten viel ungenauer ist, als es bei den Strategiewerten der Fall ist. Da die Charakterwerte auch über einen längeren Zeitraum betrachtet nur Schätzungen sind, wäre eine Betrachtung auf operativer Ebene, also innerhalb einer Handrunde, wenig sinnvoll.

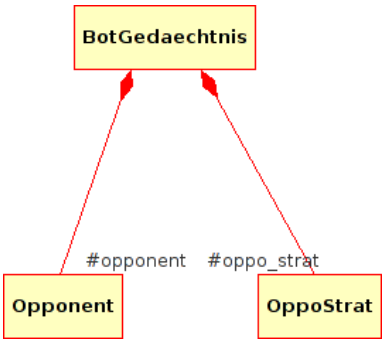



**Abbildung 7:** Die Klassenstruktur der opponent-modeling Klassen

### 3.7.4 Klassenstruktur

Für die Charakter- und Strategiewerte wurden nun jeweils eine Klasse implementiert, die sich ausschließlich um das Halten und Verrechnen der ermittelten Werte kümmert. Dies ist einerseits die Klasse „Opponent“ für die Charakterwerte und andererseits die Klasse „OppoStrat“ für die Strategiewerte. Für das Ermitteln der tatsächlichen Werte ist derweil die Klasse „BotGedaechtnis“ zuständig. Wie der Name bereits nahelegt, stellt diese Klasse das Gedächtnis für einen Bot im Spiel dar. Jeder der KI-Spieler hält also

eine Instanz dieser Klasse. Innerhalb der Klasse wiederum existiert für jeden potentiellen Gegenspieler (bis zu sechs in PokerTH) eine Instanz der Verwaltungsklassen „Opponent“ und „OppoStrat“, realisiert in Form eines Arrays.

### 3.7.5 Spielverhalten

Die Art und Weise, wie ein Spieler spielt, lässt sich an der Ausprägung zweier Dimensionen festmachen, wobei gute Spieler ihr Spiel variieren.

Die erste Dimension enhält die Extreme:

- *Loose*
- *Tight*

und die zweite Dimension spaltet sich in:

- *Passive*
- *Aggressive*

Die erste Ausprägung ist auf das Geldausgeben gemünzt:

Der loose Spieler ist sehr freigiebig mit seinem Geld, wo hingegen der tighte Spieler eher zurückhaltend und diszipliniert mit diesem umgeht.

Die zweite Ausprägung beschreibt eher die Aktivität des Spielstils:

Wo der passive Spieler eher weniger aktive Aktionen vollzieht (z.b. Call/Check), übernimmt der aggressive Spieler mehr die Initiative.

Aus diesen zwei unterschiedlichen Dimensionen, lassen sich per Kombination nun Spielertypen spezifizieren:

- *Loose-Passive*
- *Loose-Aggressive*
- *Tight-Passive*
- *Tight-Aggressive*

Der Loose-Passive Spieler ist ein Spielertyp, der fast jede Hand spielt - sei sie gut oder schlecht - jedoch kaum aktiv erhöht. Selbst gute Blätter werden in der passiven Spielweise angespielt. Grund für diese Spieler ist, dass beim Texas Hold'em natürlich auch die schlechteste Starthand, gute Gemeinschaftskarten vorrausgesetzt, zum Gewinn des Pots führen kann.

Auch der Loose-Aggressive spielt fast jedes Blatt, jedoch verfolgt er eine andere Strategie. Durch den hohen Geldeinsatz seinerseits und das andauernde unter Druck setzen durch sogenannte bets versucht dieser Spieler den Pot zu erlangen.

Ganz im Gegenteil zum Tight-Passive Spieler. Dieser spielt nur die besten Hände an, wohl weißlich, dass gute Hände eine deutlich bessere Möglichkeit zum Gewinn des Pots bieten. Jedoch lässt er sich von den Anderen nicht aus der Ruhe bringen und tut nur seinen Teil dazu bei, im Spiel zu bleiben.

Sitzt jedoch ein Tight-Aggressive in der Runde, ist anzunehmen, dass dieser auch nur die allerbesten Hände spielt, mit diesen jedoch viel Geld verdienen möchte, indem er selbst viel Geld in den Pot hineingibt.

Um diese divergenten Einschätzungen zu bewerkstelligen müssen zusätzliche pokerspezifischen Werte eingeführt werden, die dies ermöglichen.

- *VPIP* - Voluntarily Put Money Into Pot
- *PFR* - Percentage of Pre-Flop Raises
- *AF* - Post-Flop Aggression
- *WtSD* - Went to Showdown

Der VPIP speichert nun alle Aktionen, indem Geld in freiwilliger Entscheidung, sei es durch ein bet/raise/reraise oder durch einen call, in den Pot hinzugefügt wurde. Ein check oder das Setzen eines Blinds wird hier nicht berücksichtigt. Anschliessend errechnet sich ein Prozentwert, anhand aller getätigten Aktionen im Nenner.

Der PFR speichert in wieviel Prozent der Fälle der Spieler in der Phase vor dem Flop einen raise getätigt hat.

Zum AF gibt es verschiedene leicht unterschiedliche Definitionen. Es zählen aber immer nur Post-Flop Werte. Ein Beispiel für eine Definition ist die Aggression mit der Formel: $(raise\% + bet\%)/call\%$ Die verwendete Definition ist jedoch ein Wert analog zum PFR, welcher allerdings für Spielaktionen gilt, nachdem der Flop aufgedeckt wurde.

Der WtSD zeigt an wie oft (prozentual) ein Spieler bis zum Aufdecken der Karten, dem Showdown, geblieben ist.

Die zugeschnittene und auf unser Problem anwendbare Einschätzung der Spielercharaktere, ist nun noch ein wenig ausführlicher und verzichtet sogar auf einen, anfänglich geplanten, zu berechnenden Wert.

Ein Spielercharakter enthält dann z.B. folgende Beschreibung *Semi-Loose Passive/Aggressive* mit den Wertebereichen

- VPIP: 20-30
- PFR: $<5$
- AF: $>2$

dies bedeutet nun, dass der Spieler weder Loose noch Tight spielt, sich also genau zwischen beiden Extremen bewegt, in der Pre-Flop-Phase eine passive und in der Post-Flop-Phase eine aggressive Strategie verfolgt. Zur weiteren Einsicht befindet sich die Tabelle im Anhang L auf Seite 166.

Nun ist es so, dass das Verhalten eines Spielers sich nicht nur anhand seiner Karten unterscheidet, sondern von Zeit zu Zeit wechselt, aber trotzdem eine charakteristische Prägung erhält. Die Gründe hierfür können Anpassung an die Gegenspieler ebenso wie auch der emotionale Zustand sein. Um diesem zu begegnen und in Anlehnung an ein menschliches Kurz- und Langzeitgedächtnis haben wir uns für folgende Clusterung der Werte entschieden:

- *strategisch*
- *taktisch*
- *operativ*

Die Sinnhaftigkeit dieser Aufschlüsselung ist schnell zu erkennen. Während die strategische Sicht die eher langfristige Einschätzung des Charakters bietet, erlaubt die taktische Komponente mögliche mittelfristige Taktiken des Spielers zu erkennen und die Anpassung des Spielers an den Tisch mitzukalkulieren. Der operative Einblick kümmert sich um die aktuelle Hand des Spielers und erreicht somit eine recht kurzfristige Interessenseinschätzung.

Neben der Modellierung des Spielercharakters, kann nun auch geeignet auf dynamische Spielverhaltensänderungen reagiert werden. Die zusätzliche operative Erfassung kann Hinweise auf Bluffs und Slowplay liefern.

### 3.7.6 Speichermodell

Um eine spielergerechte Einschätzung zu bekommen, wäre es komfortabel auf die Kalkulierungen der letztgespielten Runden einer Person zugreifen zu können. Dafür gibt es verschiedene Ansätze. Zum einen kann ein charakteristischer Peak-Wert im Verhalten abgespeichert werden. Dies verringert die Anzahl zu speichernder Werte im Gegensatz zum allumfassenden Speichern erheblich. Jedoch können aus Peak-Werten nur schwer alle Seiten eines Charakters erfasst werden, zudem erscheint auch eine Auswahl an Werten als schwierig. Der Verlust von Informationen kann zu einer späteren Fehleinschätzung führen, daher wurde entschieden diese Option nicht zu wählen. Auch wäre es möglich gewesen, sofern nun mehrere Runden zur Verfügung stehen, die Berechnung eines Mittelwertes der Runden zu forcieren. Dies wird in einem anderen Kontext noch als wichtig erscheinen. Der Vorteil des Verfahrens wäre eine Glättung der Werte. An der Speichermenge aller Runden ändert sich hier jedoch nichts.

Mögliche Runden, wie im vorangegangenen Absatz beschrieben, können nun

- *Handrunde*
- *Bietrunde*, oder
- *Teilbietrunde*

sein. Eine Teilbietrunde ist daher eine Runde, in der der Spieler, bei Tisch, sofern er noch nicht gefolded hat, genau eine Aktion (z.B. call/bet/fold) abzuschliessen hat, bis er wieder an einer Aktion teilhaben muss. Die Bietrunde ist nun eine beliebige Anzahl, nur begrenzt durch den Cash, von Teilbietrunden. Nach einer Bietrunde tritt das Spiel in eine neue Spielphase (z.B. Flop, River, Turn, Showdown) ein. Sind nun die fünf Bietrunden abgehandelt ist eine Handrunde vollendet und neue Handkarten werden ausgeteilt.

Eine sinnvolle Zusammenführung von der Verhalten-Sicht zu den Runden ist an eine Notwendigkeit der Datenspeicherung gekoppelt. Um eine Bietrunde zu berechnen ist es notwendig alle Teilbietrunden zu kennen. Ebenso sind für die Handrunde die Bietrunden essentiell.

Die operativen Werte beinhalten nun bis zu fünf Runden einer Teilbietrunde, abgelegt in einer Queue. Anschliessend werden die überlaufenden Werte, sofern es mehr als fünf Bietrunden gibt, in einem Gesamtwert zusammengefasst, der aus einem kummulierten Wert und einem Zähler besteht. Ist eine Bietrunde vorüber, werden alle Teilbietrunden zusammengerechnet und in die bis zu 10 Bietrunden fassende Taktik-Queue eingefügt. Ist diese einmal gefüllt, müssen Werte in den Strategiewert, ein Wert der alle bisherigen taktischen Werte inkludiert, ausgelagert werden.

Aus strategischer Sicht gibt es nun zwei Wege zu beschreiten. Entweder der Charakterwert wird umgehend direkt einem Update unterzogen, sofern eine neue taktische Handrunde zur Verfügung steht oder der Wert wird erst bei einem Queue-Überlauf berechnet. Direktes Update bietet den Vorteil der Aktualität des Charakterwertes, zu dem Preis der Schwankung von Werten. Dies könnte, auf Kosten der Aktualität, mit einem Mittelwert über die Queue eingedämmt werden.

Bei Beiden ist die Möglichkeit gegeben die letzten Handwerte (in der Queue) aus dem Charakterwert heraus- oder hineinzurechnen. Wir haben uns für die direkte Variante entschieden.

Sind nun die Daten der Speicherung klar strukturiert, entsteht die Frage nach der Dauerhaftigkeit der Datensammlung. Wir haben uns, in Anlehnung zum Menschen, dafür entschieden eine Erinnerung einzuführen, die den Spieler betrifft. D.h. hat ein Bot schonmal gegen diesen Spieler gespielt, können strategische Werte zur Einschätzung des Spielers geladen werden, statt jegliche Information zu vergessen.

Diese Modellierung birgt aber ein Problem. Sofern es der Wille ist immer mit den selben Bots zu spielen, wird das Gedächtnis zweier unterschiedlicher Bots über den selben Gegner derart einander gleichen, wie es unter Menschen nicht der Fall wäre.

Der Einführung von Verfälschungswerten mit Hilfe eines botspezifischen Faktors, dem *Anpassungswert*, ist es zu verdanken, dass dieses Szenario nur noch in der theoretischen Möglichkeit, gleicher Zufallszahlen, oder Gleichheit wegen statistisch langer Spielzeit, besteht.

Um eine gleichzeitige Normierung auf [0,1] zu erwirken, wird folgende Formel verwendet:

$$\max\{0,\min\{1,\ ((1\text{-anpassungswert})*\text{random}()+0.5+\text{anpassungswert}/2)*\text{wert}\ \}\ \}$$

wobei random() in [0,1] liegt. Ist der Anpassungswert 1, wird der Wert ohne Modifikation gespeichert. Ist er jedoch 0, kann es eine maximale Abweichung vom 0,5- oder 1,5-fachen des Wertes geben.

### 3.7.7 Speicherfunktion

Da, per strategischer Sicht, keine Endlos-Queue ratsam erscheint, muss jeder Wert in wenigen Variablen abgehandelt werden. Sinnvoll dafür erscheinen zwei Zähler. Der Eine zählt den Wert, der Andere liefert die Anzahl der Handrunden, durch die geteilt werden muss, um einen Prozentwert zu erhalten.

Eine wirklichkeitsgetreue Kalkulierung des zu zählenden Wertes erscheint somit stark von der anzuwendenden Funktion abhängig. Für diese stehen zur Verfügung

- exponentiell
- linear und
- konstant.

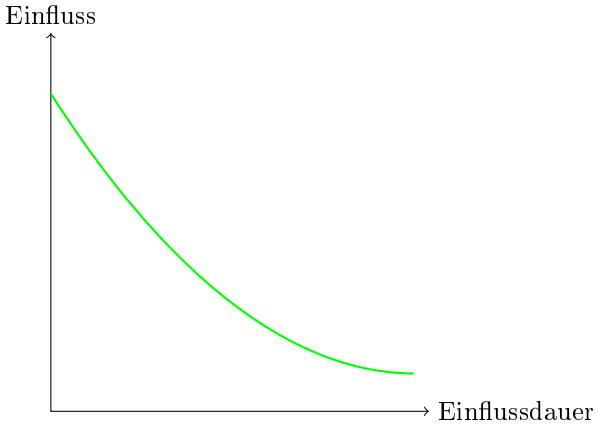



**Abbildung 8:** exponentielle Funktion

Die exponentielle Funktion, wie in Bild 8 veranschaulicht, würde sich nach der Formel

$y/x$ * neuerWert + $(x - y)/x$ * alterWert

berechnen und bietet den Vorteil, nur einen Wert speichern zu müssen. Dies bietet gleichzeitig einen grossen Einfluss (je nach Wahl von x und y) auf den Charakterwert, also einer dynamisch schnellen Anpassung vom diesem.

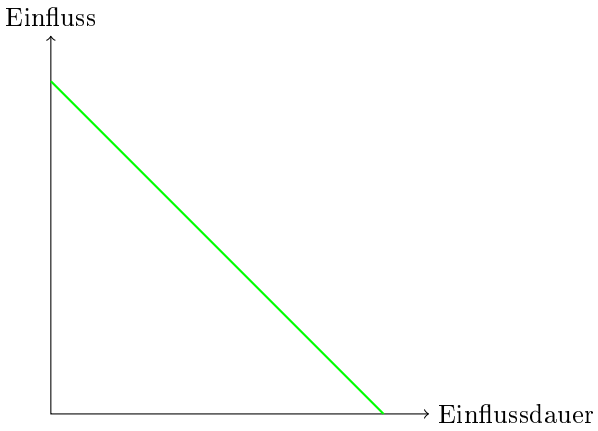



**Abbildung 9:** lineare Funktion

Die lineare Funktion, wie in Grafik 9 illustriert, berechnet sich nach der Formel

($y/x$ * neuer Wert + Rest)/$(1/2 + x/2)$

indem y einen Zähler darstellt, der anfänglich x gleicht. Jede neu eingestellte Handrunde wird y dekrementiert, bis es 0 erreicht und der damalige Wert keinen Einfluss mehr ausübt. Der Nenner $1/2+x/2$ ist hierbei ein Normalisierungswert, um eine Normalisierung auf das Intervall [0,1] zu bewirken. Der sogenannte Rest stellt die anderen „neuerWert“ dar, welche zu einem früheren Zeitpunkt in die Formel eingegangen sind, dessen y aber noch nicht 0 erreicht hat.

Der Nachteil der Speicherung zusätzlicher Einzelwerte bietet die Möglichkeit, dass sich ein Charakter alle x Handrunden komplett ändern kann, also keine Altlasten aus früheren Spielen erscheinen. Auch eine schnelle Anpassung der Werte wird geboten, wobei die Frage zu behandeln wäre, ob und wie schnell Charaktere sich so grundlegend ändern können.

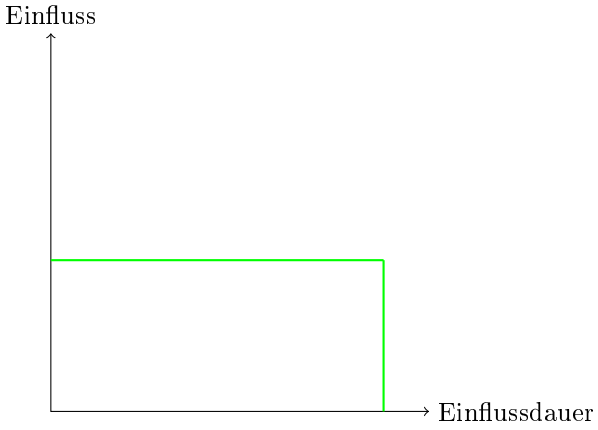



**Abbildung 10:** konstante Funktion

Die konstante Funktion, wie in Abbild 10 verdeutlicht, wird berechnet durch

$1/x$ * neuerWert (bleibt für x Runden) + Rest

Rest stellt hierbei ältere „neuerWert“ dar. Der neue Wert bleibt hier für x Runden aktuell, bis er entsorgt wird.

Auch hier müssen x Werte gespeichert werden, zu der Belohnung oder dem Preis der schnellen dynamischen Anpassung des Charakterwertes, ohne Altlasten. Die letzten x Werte sind somit gleich viel Wert, was eine Minderung des anschliessenden Problems bewirkt.

Die Frage nach der Art der Umformung vom taktischen zum strategischen Wert fiel zugunsten der exponentiellen Funktion. Bei allen drei Funktionen besteht jedoch das Bluffproblem.

### 3.7.8 Bluffproblem

Sofern ein Bluff eines Spielers erkannt wird, tritt dieses Problem auf. Für den bluffenden Spieler wird also aktuell ein Wert (z.B. Ehrlichkeit) gesetzt. Dieser erhält jedoch, bedingt durch seine Aktualität, zuviel Einfluss auf die Einschätzung, ob dieser Spieler erneut bluffen wird. Es ist anzunehmen, dass ein Spieler, der ab und an einmal einen taktischen Bluff setzt, nicht in der Folge direkt wieder blufft, um anschliessend nochmal und immer weiter zu bluffen. Dies wird jedoch vom System so angenommen.

Ein kleines Beispiel zur Anschaulichkeit:

Annahme: Spieler $A$ blufft in 30% der Fälle,
dies ist auch so bekannt

Zur Zeit $n$ führt $A$ nun einen Bluff aus.

Zeit

$n$ Bluffwahrscheinlichkeit = 30% - Spieler blufft (wird erkannt)

$n+1$ Bluffwahrscheinlichkeit = 60% - Spieler blufft nicht (Peakwert)

$n+2$ Bluffwahrscheinlichkeit = 45% - Spieler blufft nicht

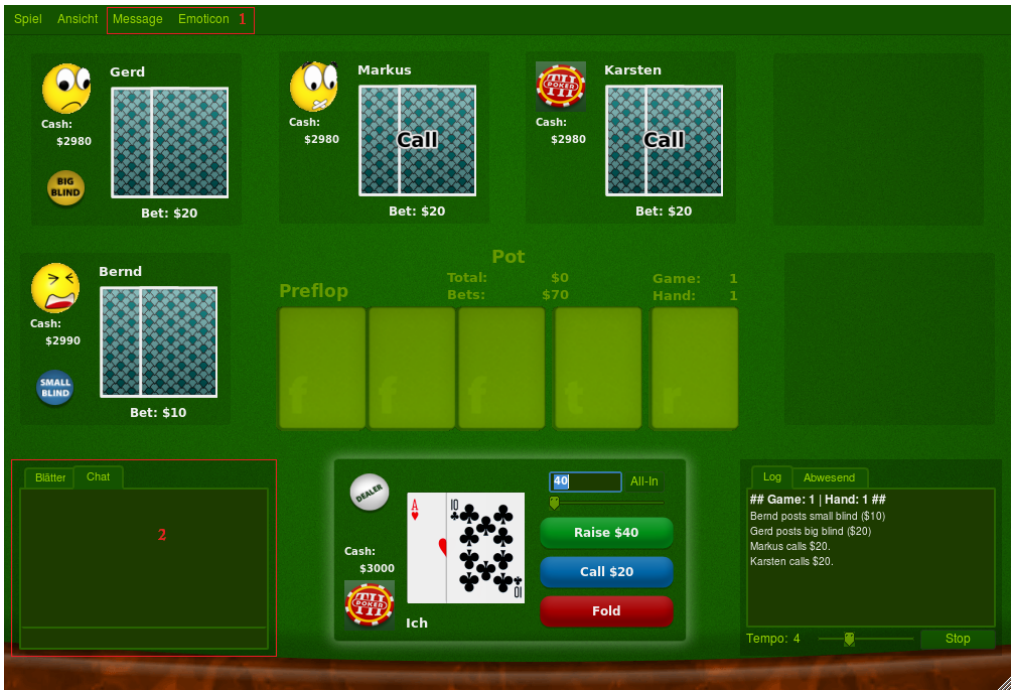



**Abbildung 11:** Modifiziertes PokerTH Bild 1, 1: erweiterte Menüleite, 2: Chatbox

Nun ist es zwar richtig zu erwarten, dass jemand nun häufiger bluffen wird, aber nicht in so einem kurzzeitigen Kontext. Der Peak besagt nun, dass er in Runde $n+2$ mit 60% wieder bluffen wird, was bei 30% Grundannahme nur eine Eintrittswahrscheinlichkeit von 30% hat.

Eine Lösung für das Bluffproblem wurde nicht erarbeitet und folglich nicht in den Code übertragen.

## 3.8 Erweiterungen

### 3.8.1 GUI

Um dem Ziel der verbalen und non-verbalen Kommunikation zur Modifikation der Emotionen der NPCs innerhalb der PokerTH Umgebung näher zu kommen, musste die grafische Benutzeroberfläche (engl. „**G**raphical **U**ser **I**nterface“) erweitert werden.

Es wurden zwei weitere Menüpunkte erstellt (vgl. Abbildung 11 Markierung 1), welche dem Benutzer die Möglichkeit bieten, zuerst aus einer der vier Emotionsgruppen (Freude, Schadenfreude, Wut und Nervosität, siehe Abschnitt 3.5.2 auf Seite 16) ein Emoticon auszuwählen und später eine dazu passende Nachricht an alle, also an den Tisch, oder an einen bestimmten Gegner zu verschicken. Außerdem wurde die bereits vorhandene Chatbox standardmäßig beim Spielstart in den Vordergrund verschoben (vgl. Abbildung 11 Markierung 2 sowie Abbildung 15 Markierung 1), damit auch unerfahrene Spieler die verschickten Nachrichten der NPCs sofort erkennen.

Zu Beginn des Spiels sind die einzelnen Einträge im Menü *Messages* deaktiviert. Erst durch die Auswahl eines Emoticons werden die Einträge aktiviert. Ein ausgewähltes Emoticon erscheint sofort im Profil des Spielers und zeigt somit seine ausgewählte Gemütslage an (vgl. Abbildung 13 Markierung 2). Im Messagemenü ist ab diesem Zeit-

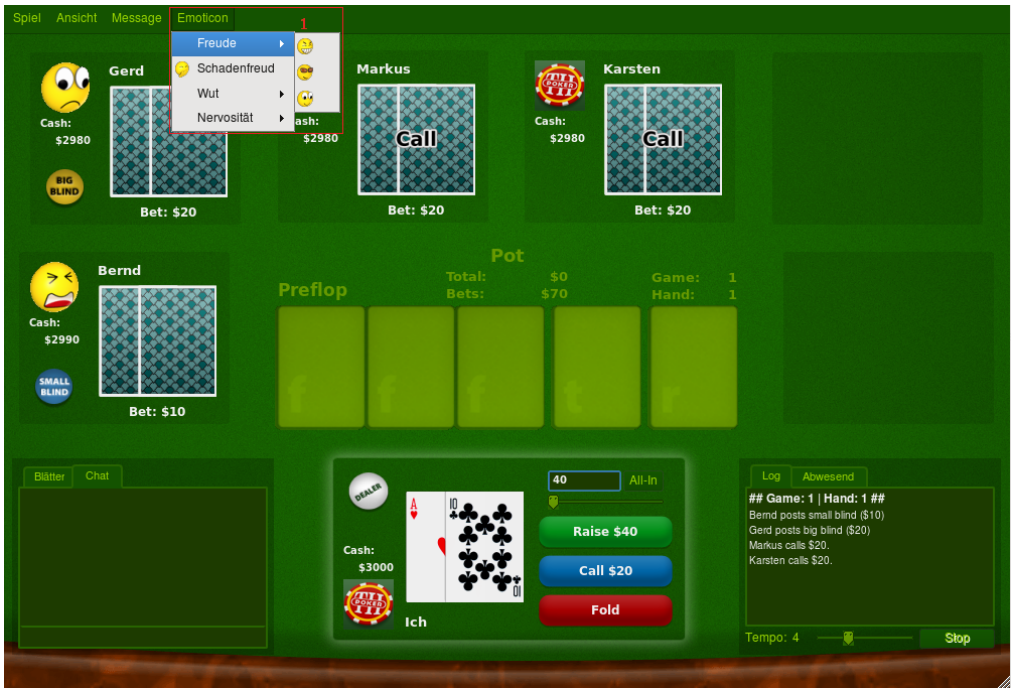



**Abbildung 12:** Modifiziertes PokerTH Bild 2, 1: Menüleiste - Auswahl eines Emoticons

punkt die Nachrichtengruppe aktiviert, die zu der Emotion des ausgewählten Emoticons passt (vgl. Abbildung 13 Markierung 1). Die Nachrichten sind auf der ersten Ebene nach Emotionsgruppen (Freude, Schadenfreude, Wut, Nervosität) unterteilt. Auf der zweiten Ebene findet noch eine Einteilung nach Empfänger statt. Der Spieler verschickt entweder eine Nachricht „*an den Tisch*“ was der erste Untermenüeintrag ist, oder richtet sie an einen bestimmten Gegner, was im Menü durch den Untermenüeintrag „*gerichtet*“ realisiert ist. Soll eine Nachricht an einen Gegner gerichtet werden, so muss in einem Popup der Empfänger ausgewählt werden (vgl. Abbildung 14 Markierung 1).

Durch die Auswahl einer Nachricht wird diese zum einen in der Chatbox angezeigt, zum anderen werden durch vordefinierte Formeln die Emotionen entweder aller NPCs, falls die Nachricht an den Tisch gerichtet wurde, oder eines bestimmten NPCs, falls die Nachricht an einen bestimmten Gegner gerichtet war, entsprechend ihrer Charaktereigenschaften und ihrer vorherigen Emotionswerte angepasst.

Realisiert wurden die beschriebenen Änderungen, indem die Datei *mainwindow.ui* mit dem QT Designer erweitert wurde.

### 3.8.2 Config.xml

In PokerTH wurde nur zwischen Menschen und Bots unterschieden. Es gab somit auch keine Möglichkeit irgendwelche Gegenspieler explizit auszuwählen. Da das Ziel der Projektgruppe allerdings darin bestand, Computerspieler mit individuellen Charaktereigenschaften und Emotionen zu generieren, musste die explizite Spielerauswahl in das Spiel integriert werden. Dies wurde mittels einer Parameterdatei, der *config.xml*, bewerkstelligt. In ihr können alle sieben potenziellen Spieler, inklusive dem menschlichen Spieler, definiert werden. Es besteht die Möglichkeit, den menschlichen Spieler – wie im normalen PokerTH – als so genannten *CITeacher* zu erstellen. Wird der menschliche Spieler als *CITeacher* erstellt, so wird eine Logdatei generiert, mit der später das Neuronale Netz lernen kann. Weiter können die Bots entweder als *LocalPlayer* (in PokerTH

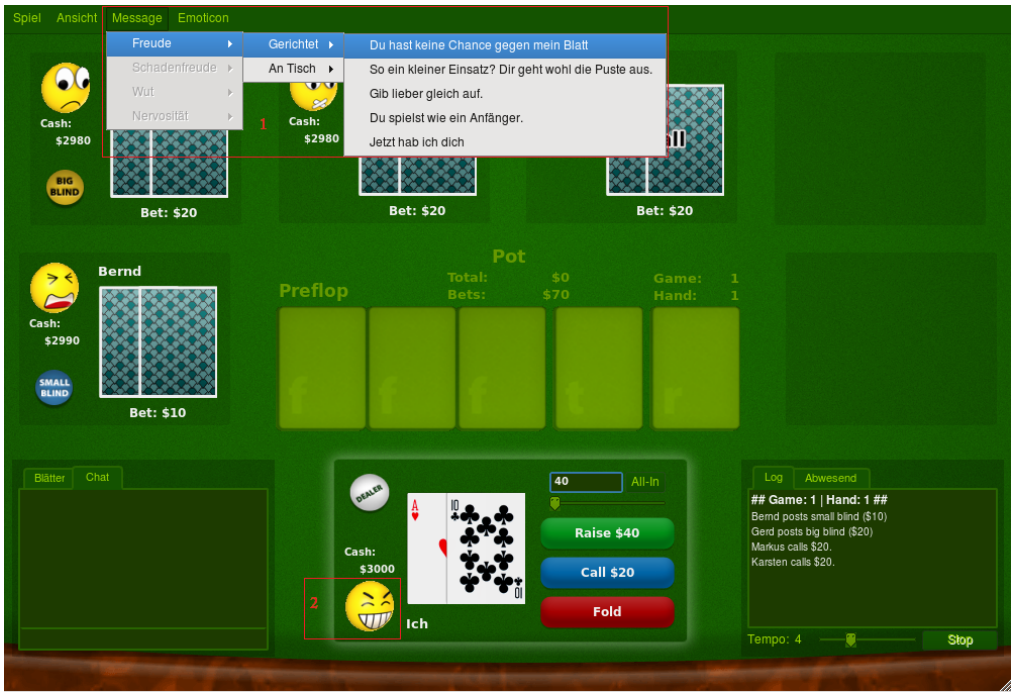



**Abbildung 13:** Modifiziertes PokerTH Bild 3, 1: Menüleiste - Auswahl einer gerichteten Nachricht, 2: gesetztes Emoticon des Spielers

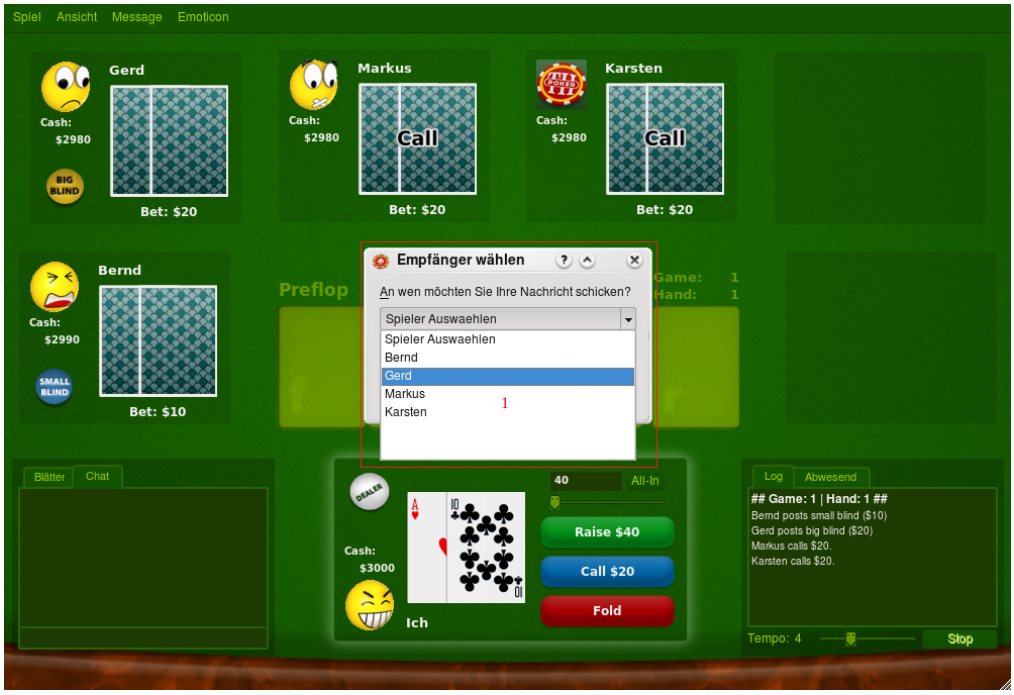



**Abbildung 14:** Modifiziertes PokerTH Bild 4, 1: Empfängerauswahl Dialog

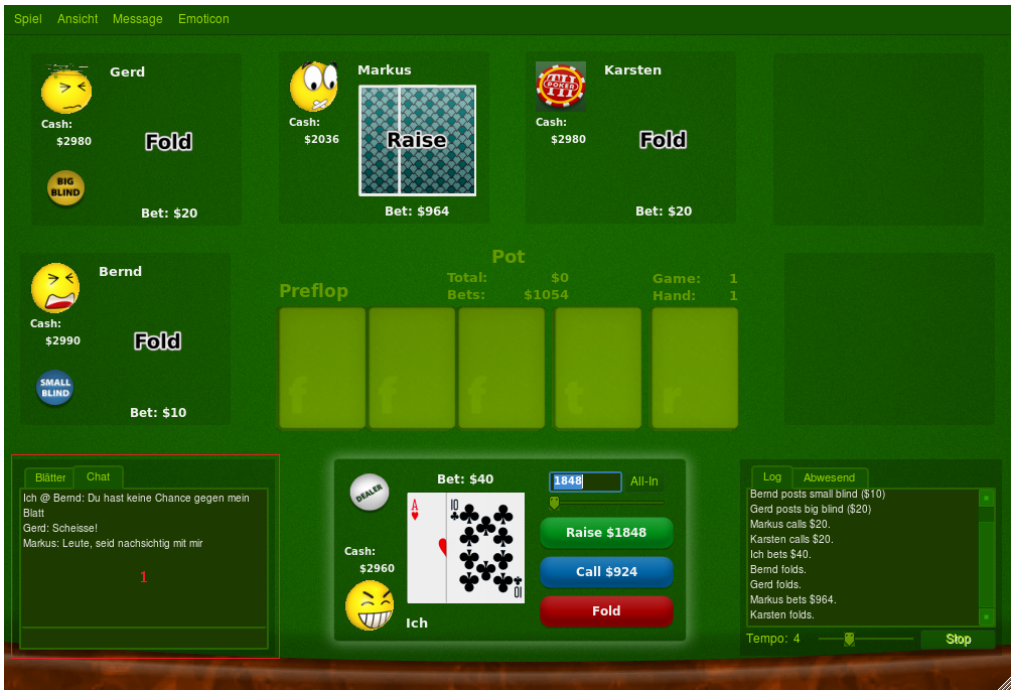



**Abbildung 15:** Modifiziertes PokerTH Bild 5, 1: Chatbox

Standart für alle NPCs bei einem lokalen Spiel) oder als *CIPlayer* erstellt werden. Als *CIPlayer* haben die Bots einen individuellen Charakter, für den die Charakterwerte (vgl. Abschnitt 3.5.1 auf Seite 15) ebenfalls in der Datei *config.xml* definiert werden, Emotionen (vgl. Abschnitt 3.5.2 auf Seite 16) und ein Botgedächtnis (vgl. 3.7 auf Seite 33), mit dem er seine Gegner einschätzen kann.

Im Laufe der Projektarbeit wurde deutlich, dass das Spielverhalten der Computerspieler bewertet werden muss. Zu diesem Zweck kann man sich die Karten der Bots permanent anzeigen lassen. Dies wird ebenfalls über die Datei *config.xml* gesteuert. Ebenso kann das Starten der Umfrage (vgl. Abschnitt 3.9 auf Seite 48), die zur Validierung der Projektarbeit durchgeführt wurde, über diese Parameterdatei eingerichtet werden.

### 3.8.3 Gamelistener

Damit Erweiterungen in das bestehende Spiel eingebunden werden können, muss gewährleistet werden, dass neu generierte Objekte Zugriff auf die Spielbewegungen erhalten. Unter der Prämisse, den Code so wenig wie möglich zu modifizieren und in der späteren Benutzung flexibel zu bleiben, wurde eine Observer-Architektur nach Gamma et al. 2004 (siehe [GHJV04]) eingerichtet. Um eine zentrale Registrierung zu ermöglichen, musste also ein Objekt gefunden werden, welches von allen zu modifizierenden Code-Stellen erreichbar ist. Die Referenzen auf die Abonnementen werden deshalb im verfügbaren *Session*-Objekt abgelegt.

Die Idee einer Observer-Architektur wurde durch den *PokerTHAdapter* realisiert. Der Publisher, bei uns die Klasse *GameListener*, informiert nun per Referenz alle Abonnementen über die relevanten Spielinformationen. Die Anmeldung erfolgt statisch über einen Eintrag in der Klasse *PokerTHInitializer*, oder durch direkte Anmeldung beim Publisher-Objekt. Um die Spielinformationen zu erhalten müssen die vom Publisher aufgerufenen Funktionen geerbt werden. Die Grafik 16 auf Seite 47 zeigt die Observer-

architektur inklusive den erbenden Klassen.

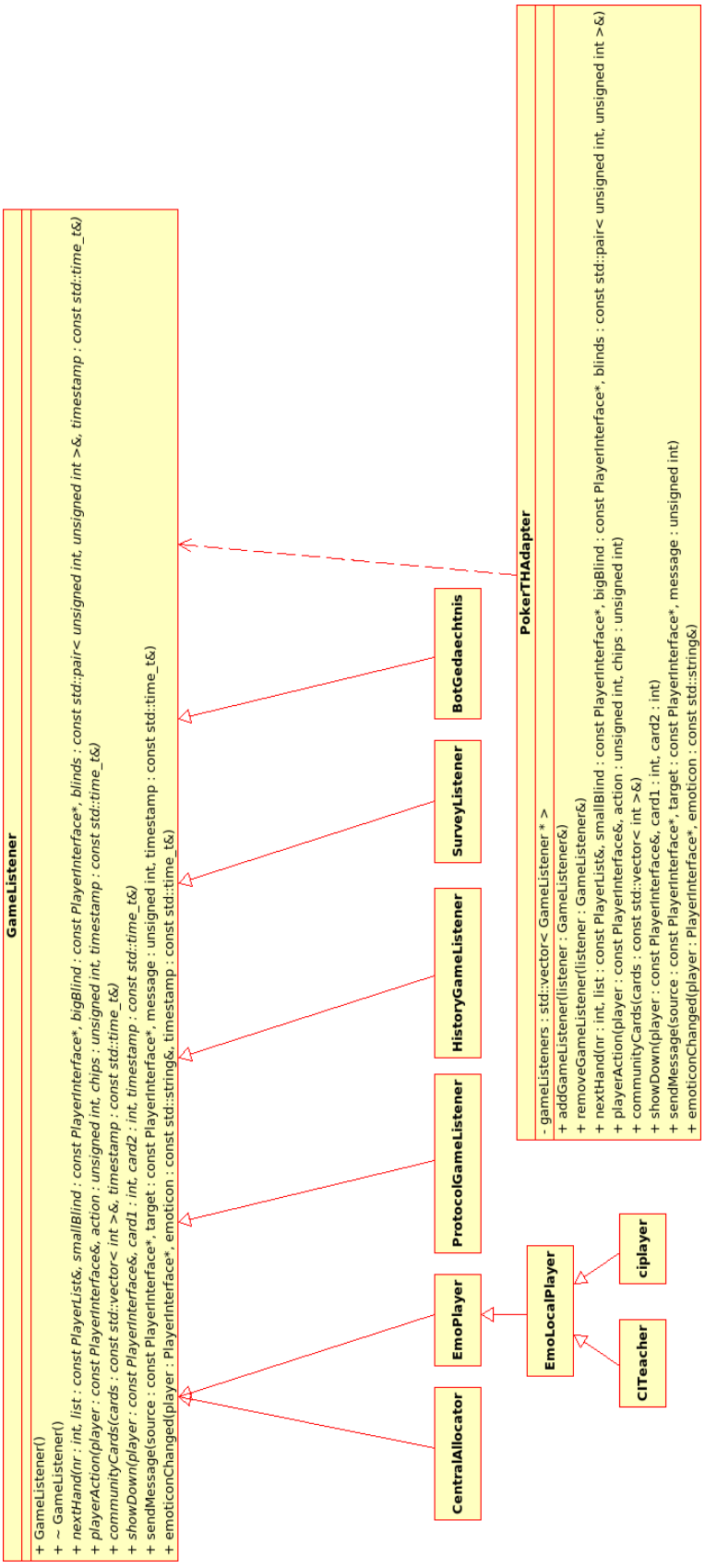



**Abbildung 16:** Klassendiagramm der Observerarchitektur, mit erbenden Klassen

## 3.9 Umfrage

### 3.9.1 Aufbau der Umfrage

Die Teilnehmer der Projektgruppe 529 hatten schon zu Beginn der Planungsphase berücksichtigt, dass die erstellten Arbeiten unter verschiedenen Gesichtspunkten validiert und verifiziert werden mussten. Aus diesem Grund war am Anfang des Semesters festgelegt worden, dass die erstellte Software mit Hilfe einer Umfrage bewertet werden sollte.

Ein Proband sollte für die Umfrage eine festgelegte Zeit das erweiterte PokerTH spielen und danach einen Fragebogen ausfüllen. Während des Spielens sollten dann von der Software alle Aktionen des Spielers gespeichert werden, um die Antworten auf dem Fragebogen später verifizieren zu können. Dieses Vorhaben wurde durch den Listener (siehe 3.8.3 auf Seite 45) realisiert.

Die Durchführung der Umfrage sollte folgende Fragen beantworten:

- Sind unsere Bots menschlicher als der schon im Spiel vorhandene Bot?
- Sind unsere Bots spielstärker als der schon im Spiel vorhandene Bot?
- Werden die Erweiterungen von den Spielern angenommen, oder wird der Spielspass dardurch geringer?
- Können die allgemeinen Vorurteile bekräftigt werden, dass weibliche Testspieler mehr kommunizieren als männliche?

Schon früh stellten die Teilnehmer fest, dass eine professionelle Beratung bezüglich Umfragen die Resultate stark verbessern könnte. Aus diesem Grund wurde mit dem Statistischen Beratungs- und Analysezentrum der Fakultät Statistik der Technischen Universität Dortmund[9], kurz SBAZ, Kontakt aufgenommen. Das SBAZ hat das Ziel, die Fakultäten, Institute und zentralen Einrichtungen der Technischen Universität Dortmund einschließlich Doktoranden und Diplomanden bei der Planung und Auswertung ihrer Umfragen und Experimente sowie bei der Optimierung ihrer Prozesse zu unterstützen.

Zur Diskussion stand am Anfang noch, ob ein elektronischer Fragebogen eingesetzt wird, oder ob die Befragung in Papierform stattfinden sollte. Da die Argumente der leichteren Auswertbarkeit und der besseren Verknüpfungsmöglichkeit mit den Spieldaten für den elektronischen Fragebogen sprachen, fiel die Entscheidung letztendlich gegen den Papierfragebogen.

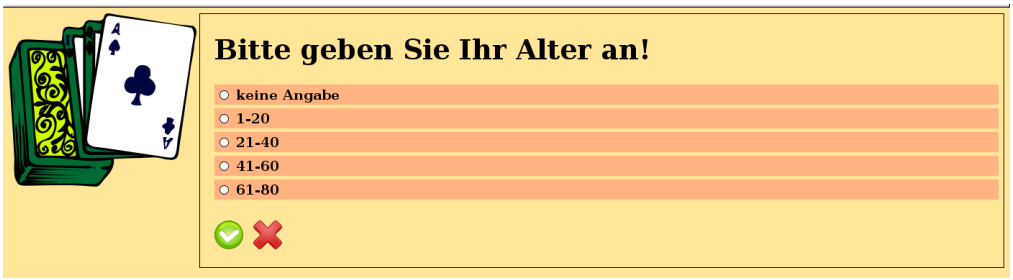



**Abbildung 17:** Teil der mit dem Spring Framework realisierten Umfrage

In einer Beratungsstunde beim SBAZ wurde der vorläufige Fragebogen besprochen. Außerdem wurden vom SBAZ noch generelle Hinweise gegeben, wie die Umfrage und

[9] http://www.statistik.uni-dortmund.de/sbaz/de/index.html

Auswertung am besten zu bewerkstelligen seien:

- Die Umfrage sollte möglichst am Campus durchgeführt werden, wobei eine wirklich repräsentative Umfrage sehr schwer zu erlangen ist. Zur Diskussion stand auch eine Umfrage in der Dortmunder Innenstadt vor einem Spielwarengeschäft. Dies wurde aber vom SBAZ mit der Begründung abgelehnt, dass die Befragten bedingt durch Alter und Schulabschluss zu inhomogen wären und es nicht möglich wäre aus den gegebenen Antworten Ergebnisse zu erzielen.
- Der Fragebogen sollte zwischen verschiedenen Bildungsständen unterscheiden. Aus diesem Grund wurde die Frage nach dem höchsten Schulabschluss in die Umfrage integriert.
- Alle Antworten im Fragebogen sollten aus ordinale Werten (ja/nein; gut/ein wenig gut/weder noch/ein wenig schlecht/schlecht) bestehen, die eine bestimmte Ordnung haben, damit die Auswertung auf einen diskreten Werteberich abzielt.
- Die Antwortmöglichkeit „keine Angabe“ sollte bei keiner Frage fehlen, damit von einem Befragten nicht eine beliebige Antwort ausgewählt wird, falls der Befragte zu einer Frage keine Antwort äußern möchte.
- Der Fragebogen sollte nur dann ausgewertet werden, wenn alle Fragen beantwortet wurden, um einen gleichbleibenden Datenbestand zu haben.
- Es sollte keine Vorauswahl für Antworten ausgewählt sein, damit die Befragten nicht beeinflußt werden.
- Da die Umfrage über den Studentenkreis hinaus geht, sollten alle Probanden mit „Sie“ angeredet werden.
- Die Probanden sollten ca. 15 Minuten spielen und direkt danach den elektronischen Fragebogen ausfüllen.

In einer anschließenden Diskussion beschlossen die Teilnehmer der Projektgruppe, dass die Umfrage sowohl am Campus als auch im privaten Umfeld der Teilnehmer durchgeführt werden soll. Die Umfrage sollte von 50 bis 100 Teilnehmern bearbeitet werden. Sehr lange wurde auch diskutiert, wie lang das Poker-Spiel genau dauern sollte, bis die Umfrage startet. Argumente gegen eine Spieldauer von 15 Minuten waren, dass sich so nur weniger Freiwillige finden ließen, als mit einer kürzeren Spieldauer. Vor allem könnten so mehr Probanden in kürzerer Zeit an der Umfrage teilnehmen. Gegen eine kürzere Spieldauer sprach vor allem das schlechte Einschätzen der Gegner. Da es ein erklärtes Ziel der Projektgruppe war, menschenähnliche Computergegner zu erstellen, wurde die Spielzeit dann bei 15 Minuten belassen, damit die Testspieler einen besseren Eindruck von den Gegnern hatten. Beschlossen wurde auch, dass in das Testspiel drei verschiedene erarbeitete CI-Charaktere integriert wurden, sowie zur Verifikation ein Bot, der bereits in PokerTH vorhanden war. Alle Probanden sollten außerdem gegen die gleichen Gegner spielen.

Der elektronische Fragebogen wurde dann mit Hilfe des *Spring Frameworks* [10] und *Maven*[11] erstellt. *Maven* ist ein fortgeschrittenes Build-Tool für Java, dass das Übersetzen, Weiterentwickeln und Testen sehr stark vereinfacht. Es setzt auf einem Webserver, genauer Apache Tomcat, auf, der die gesamte Umfrage mit nur einer Datei wiedergeben kann. In der Umfrage-Datei befinden sich dann sämtliche Informationen, der für die Umfrage benötigten Plugins und Pfade bezüglich der Fragen der Umfrage. Als Ergebnis einer Umfrage wurde eine XML-Datei automatisch erzeugt, die alle im Fragebogen ausgewählten Antworten enthielt. Für eine Auflistung aller Fragen und Antwortmöglichkeiten: siehe K.1 ab Seite 162.

[10] `http://www.springframework.org/`
[11] `http://maven.apache.org/`

In PokerTH wurde ein Timer eingebaut, der eine Spielsession nach 15 Minuten beendet und im Webbrowser die Umfrage startet. Während des Spiels wurden alle Aktionen der Spieler in einer Log-Datei gespeichert. Diese wurde mit einer SessionID versehen, welche dann beim Aufruf des Webservers an die Umfrage weitergegeben wurde, womit im Nachhinein jeder Umfrage-Datei die passende Spiele-Log-Datei zugeordnet werden konnte.

Um den Ablauf sowie die Fragen der Umfrage zu testen wurde entschieden, dass in einem kleinen Test innerhalb der Fachschaft Informatik das Spiel sowie die Umfrage mit wenigen Probanden untersucht werden sollten. Diese erste Testumfrage zeigte, dass ein paar Fragen nicht von allen Teilnehmern richtig gedeutet wurden, weshalb der Fragenkatalog noch einmal überarbeitet wurde. Aufgrund von konstruktive Kritik seitens der Befragten wurde auch über die grafische Oberfläche sowie die multimediale Wiedergabe von PokerTH diskutiert. Die Kritik wurde darin geäußert, das eine Gruppe von Befragten das Spiel lieber mit einer Audioausgabe spielen würden und die andere Gruppe nicht. Außerdem wurde bemängelt, dass man beim Auswählen einer Nachricht oder eines Emoticons das Spielgeschehen nicht weiter verfolgen kann, da das Menü zu weit von den Karten entfern sei. In einer PG-Sitzung wurde dann beschlossen, dass alle Testspiele mit einer Audio-Ausgabe stattfinden, die Auswahlmöglichkeiten für Nachrichten und Emoticons aber, aufgrund des großen Arbeitsaufwandes, nicht mehr verändert werden. Aus diesem Grund wurde ein paar Tage später die abschließende Umfrage gestartet.

In Zweiergruppen gingen die Projektgruppenteilnehmer über den Campus oder durch verschiedene Lehrstühle und Fachschaften und baten verschiedene Personen um ihre Teilnahme. Allen Teilnehmern wurde zu Beginn kurz erklärt, dass es sich um ein modifiziertes Poker-Spiel nach den Regeln der Variante Texas Hold'em handelt. Gezeigt wurde auch, wie man seine eigenen Emotionen im Spiel durch Emoticons und Nachrichten darstellt. Um die Ergebnisse nicht zu verfälschen wurden den Probanden erst nach dem Spiel und der Umfrage auf Anfrage mitgeteilt, was genau das Ziel der Poker-Implementierung war.

Zum Ende dieser Befragung hatten 67 Teilnehmer die Umfrage bearbeitet. Die Umfrage Ergebnisse wurden vom XML-Format in ein CSV Format konvertiert.

Zur Auswertung der Umfrage wurden die Programme OpenOffice[12] sowie R[13] benutzt, womit diverse Diagramme und eine Korrellationsmatrix erstellt wurden. Außerdem wurden mit dem *Apriori*-Algorithmus für verschiedene Implikationen die Support- und Konfidenzwerte berechnet.

Im Folgenden werden die Ergebnisse beschrieben und bewertet:

[12] `http://www.openoffice.org/`
[13] `http://www.r-project.org/`

### 3.9.2 Diagramme

#### Die Teilnehmer

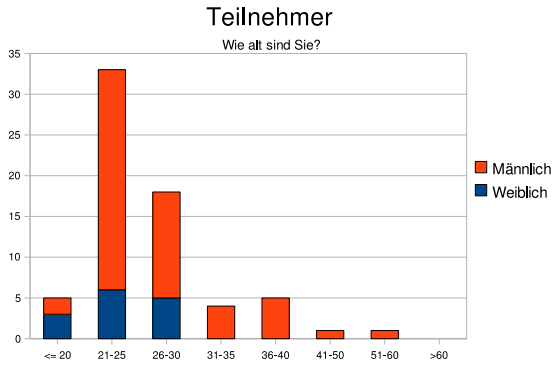



**Abbildung 18:** Alter und Geschlecht der Teilnehmer

In Abbildung 18 wird die Anzahl der Teilnehmer nach ihrem Alter und ihrem Geschlecht aufgeschlüsselt.
Hauptsächlich wurden Männer im Alter zwischen 21 und 25 befragt. Der hohe Anteil an Teilnehmern zwischen 21 und 30 Jahren erklärt sich daraus, dass die Umfrage zu einem großen Teil am Campus der Universität Dortmund, insbesondere in Cafeterien und Fachschaftsräumen verschiedener Fakultäten, durchgeführt wurde.

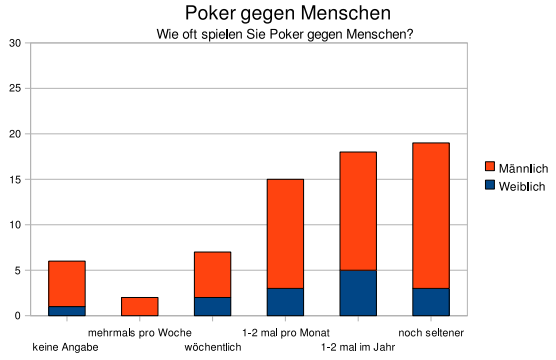



**Abbildung 19:** Wie oft spielen die Befragten Poker gegen Menschen?

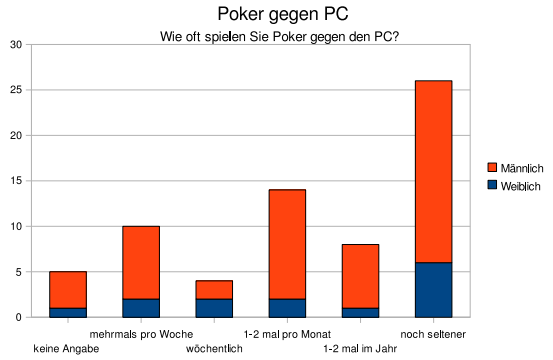



**Abbildung 20:** Wie oft spielen die Befragten Poker gegen einen Computerspieler?

Die Diagramme 19 und 20 zeigen, wie oft die Befragten Poker gegen Menschen und wie oft sie gegen den Computer spielen. Die y-Achse gibt bei beiden Diagrammen an, wie häufig eine Antwort gegeben wurde.

Leider wurden viele Menschen befragt, die nur sehr selten Poker spielen. Insgesamt gaben 20 von 67 Befragten an, noch nie Poker gespielt zu haben. Diejenigen, die Poker spielen, scheinen dafür häufiger gegen den Computer zu spielen als gegen Menschen. Dies könnte auf den in letzter Zeit großen Erfolg von diversen online Pokerräumen zurückzuführen sein.

### Die Bewertung der Computerspieler

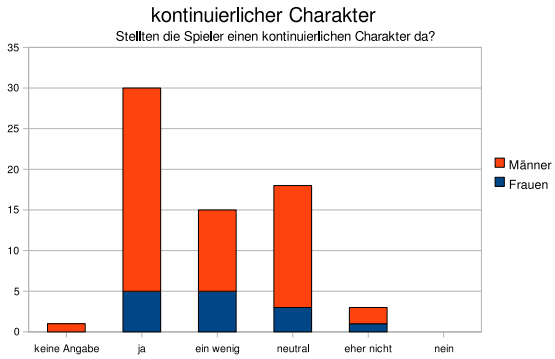



**Abbildung 21:** Stellten die Spieler einen kontinuierlichen Charakter über den Spielverlauf hinweg da?

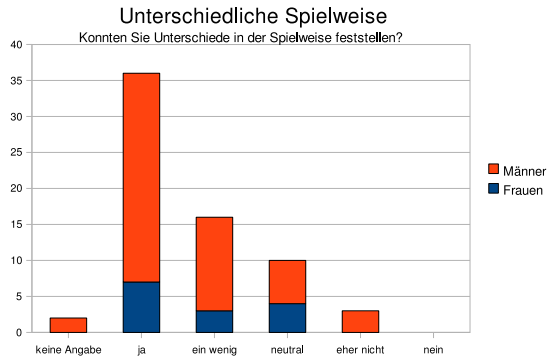



**Abbildung 22:** Wurden Unterschiede in der Spielweise festgestellt?

Auf den Diagrammen 21 und 22 kann man sehen, wieviele der Befragten eine bestimmte Antwort auf die Fragen "Stellten die Spieler einen kontinuierlichen Charakter da?" und "Konnten Sie Unterschiede in der Spielweise feststellen?" gegeben haben.

Erfreulich in Hinsicht auf das angestrebte Ziel der Projektgruppe ist, dass es eine eindeutige Tendenz dazu gab, dass die Befragten sowohl individuelle Charaktere wie auch individuelle Spielweisen bei den Pokerbots festgestellt haben. Jeder Teilnehmer spielte gegen drei aus dem Projekt hervorgegangene Computerspieler und gegen einen Bot, der standardmäßig in *PokerTH* enthalten ist.

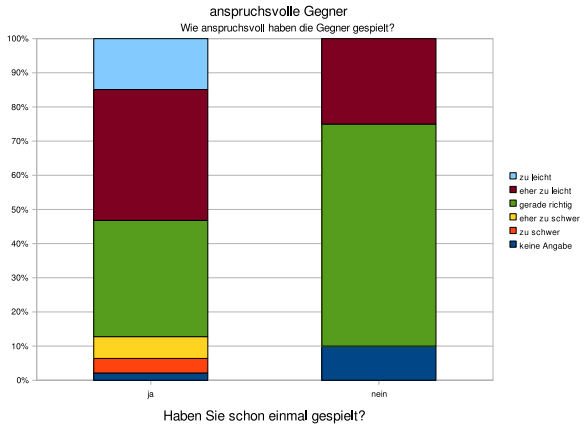



**Abbildung 23:** Wie einfach oder schwer waren die Computergegner?

Die Abbildung 23 verdeutlicht, wieviel Prozent der Befragten die Gegenspieler als *zu leicht*, *eher zu leicht*, *gerade richtig*, *eher zu schwer* und *zu schwer* empfanden. Dabei werden die Teilnehmer danach unterteilt, ob sie schon einmal Poker gespielt haben oder nicht.
Wie auf dem rechten Balken sehr schön zu erkennen ist, wurde der Schwierigkeitsgrad von denjenigen, die noch nie Poker gespielt haben, überwiegend als *gerade richtig* empfunden. Für die Befragten, die angegeben haben, dass sie schon einmal gespielt haben, waren die Computergegner hauptsächlich *zu leicht* oder *eher zu leicht*.

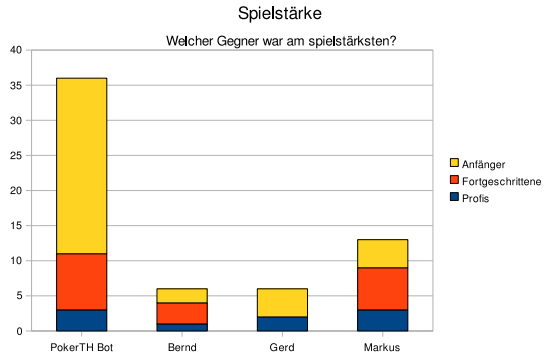



**Abbildung 24:** Welcher Gegner war am spielstärksten?

Abbildung 24 zeigt, welcher Gegenspieler von den Befragten als am spielstärksten angesehen wurde. Unterteilt ist das Diagramm weiterhin nach Spielertypen *Anfänger*, *Fortgeschrittene* und *Profis*, zu denen sich die Befragten selber zugeordnet haben.

Man erkennt sehr leicht, dass der spielstärkste Gegner der PokerTH-Bot ist. Dies kommt allerdings dadurch zu stande, dass gerade diejenigen Befragten den PokerTH-Bot als am spielstärksten ausgewählt haben, die sich selber als *Anfänger* bezeichnet haben. Vergleicht man jedoch die *Fortgeschrittenen* und die *Profis* so erkennt man, dass der PokerTH Bot sowie der CI-Player Markus fast gleich aufliegen. Begründen kann man dieses Ergebnis sehr einfach: Befragte, die noch sehr unerfahren sind, was das Spiel Poker angeht, sind der Meinung, dass die Spielstärke etwas mit einem Sieg oder einer Niederlage zu tun hat. *Fortgeschrittene* oder *Profis* wissen aber, dass auch sehr gute Spieler nicht immer gewinnen. Diese Gruppe der Befragten achtete mehr auf die Aktionen der Gegner als nur auf die entgültigen Resultate.

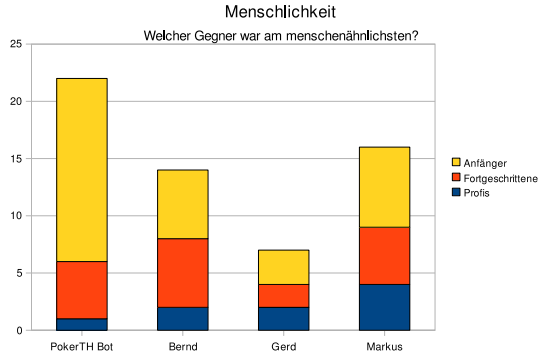



**Abbildung 25:** Welcher Gegner war am menschenähnlichsten? Aufgeteilt nach Anfänger/Fortgeschrittene/Profis.

Das Diagramm 25 zeigt, welcher Gegner als am menschlichsten ausgewählt wurde. Unterteilt ist das Diagramm wieder nach Spielertypen *Anfänger*, *Fortgeschrittene* und *Profis*, zu denen sich die Befragten selber zugeordnet haben.

Auch hier sticht zwar auf den ersten Blick hervor, dass der PokerTH-Bot insgesamt als am menschlichsten gilt, doch auch dies wird einzig und allein durch die große Anzahl an Befragten deutlich, die sich als eher ungeübte Pokerspieler bezeichnen. Vergleicht man wieder nur die Summe der *Fortgeschrittenen* und der *Profis*, so sieht man, genau wie

in Abbildung 24, sehr eindeutig, dass der PokerTH-Bot bei geübteren Befragten weder spielstärker noch menschenähnlicher ist. Es ist sogar so, dass die CI-Player Bernd und auch Markus bei der Gruppe der geübteren Befragten deutlich menschenähnlicher sind als der PokerTH-Bot.

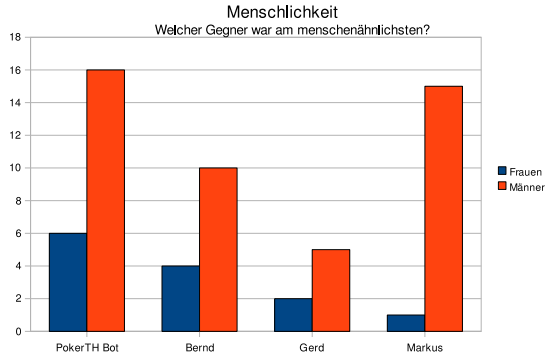



**Abbildung 26:** Welcher Gegner war am menschenähnlichsten? Aufgeteilt nach Geschlecht.

Die Abbildung 26 beschreibt, welcher Gegenspieler von den Befragten als am menschenähnlichsten angesehen wurde und unterteilt die Ergebnisse nach dem Geschlecht der Befragten.

Die befragten Damen empfanden den PokerTH-Bot eindeutig am menschenähnlichsten. Bei den befragten Herren sehen die Ergebnisse ganz anders aus. Der PokerTH-Bot teilt sich fast den Titel „am menschenähnlichsten“ mit dem CI-Player Markus. Die CI-Player Bernd liegt kurz dahinter, und der CI-Player Gerd ist sehr weit abgeschlagen.

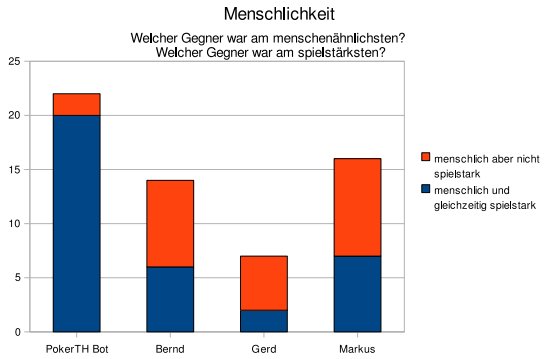



**Abbildung 27:** Welcher Gegner war am menschenähnlichsten? / Welcher Gegner war am spielstärksten?

Das Diagramm 27 stellt dar, welcher Gegenspieler am menschenähnlichsten war und unterteilt weiter danach, ob der Gegenspieler auch gleichzeitig spielstark war.

Es ist zwar wieder der PokerTH-Bot, der am menschenähnlichsten ist, aber man erkennt in diesem Diagramm ganz deutlich, dass viele der Befragten menschenähnlich mit spielstark gleichsetzten. Nur für zwei Befragte ist der PokerTH-Bot menschlich aber nicht spielstark. Vergleicht man dies mit den CI-Playern *Bernd*, *Gernd* und *Markus* so erkennt man, dass diese Gegener deutlich öfter als menschlich bezeichnet wurden, auch wenn sie nicht als am spielstärksten bewertet wurden.

### Die zusätzlichen Features

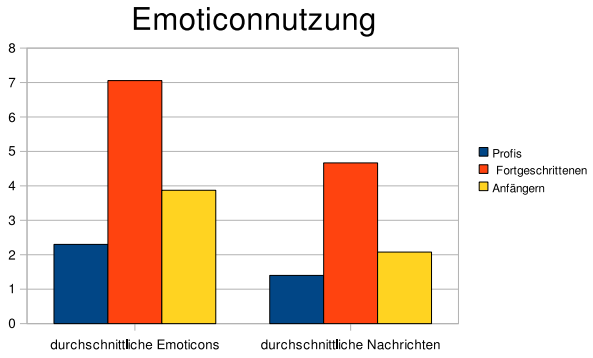



**Abbildung 28:** Wie viele Nachrichten/Emoticons wurden durchschnittlich verschickt?

Das Balkendiagramm 28 repräsentiert die durchschnittliche Emoticon- und Nachrichtennutzung, aufgeteilt nach dem eigenen Spielertyp, also *Anfänger*, *Fortgeschrittene* und *Profis*.

Generell kann gesagt werden, dass alle Befragten mehr Emoticons genutzt haben als Nachrichten. Beide Erweiterungen wurden am wenigsten von den *Profis* verwendet. Für diese Gruppe von Befragten war das Spiel eindeutig wichtiger als die erarbeiteten Erweiterungen. Darauf folgen die *Anfänger*. Für diese Gruppe war das Spiel vermutlich so einvernehmend, dass sie für Emoticons und Nachrichten keine Zeit mehr hatten. An der Gruppe der *Fortgeschrittenen* erkennt man, dass sowohl Emoticons als auch Nachrichten gut angenommen wurden. Beides wurde deutlich stärker verwendet als bei den beiden anderen Gruppen.

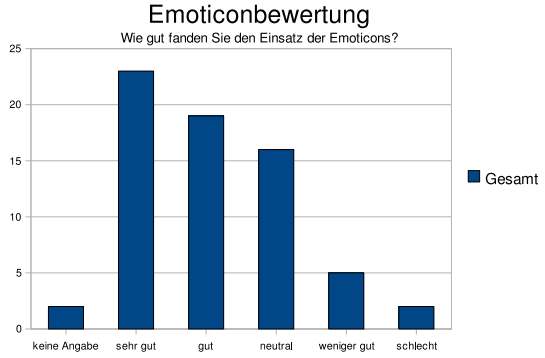



**Abbildung 29:** Wie wurden die Emoticons bewertet?

Abbildung 29 zeigt wie die einzelnen Probanden den Einsatz und die Einbindung der Emotionen im Spiel bewertet haben. Man kann erkennen, dass ein Großteil der befragten Personen diese als *sehr gut* bzw. *gut* beurteilt haben, was eine sehr hohe Akzeptanz dieses Konzeptes verdeutlicht.

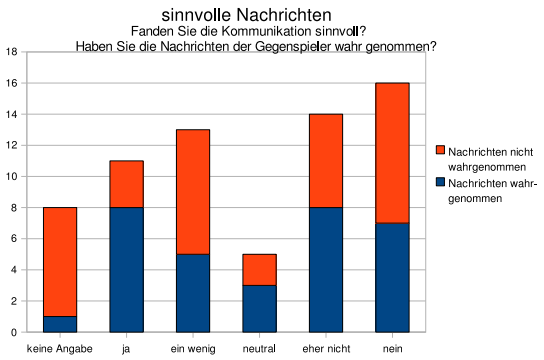



**Abbildung 30:** Wie sinnvoll waren die Nachrichten und wurden sie wahrgenommen?

Abbildung 30 zeigt, wie sinnvoll die Probanden die Nachrichten empfunden haben. In dem Diagramm werden diese Antworten noch unter dem Gesichtspunkt aufgeschlüsselt, ob die Nachrichten der Bots überhaupt wahrgenommen wurden. Die Antworten der Probanden, die die Nachrichten bemerkt haben, sollten stärkere Beachtung finden als die Antworten jener Probanden, die diese nicht wahrnahmen. Die erstgenannte Probandengruppe hat sich tendenziell mehr für *nicht sinnvoll* als für *sinnvoll* entschieden.

### Weitere Korrelationen

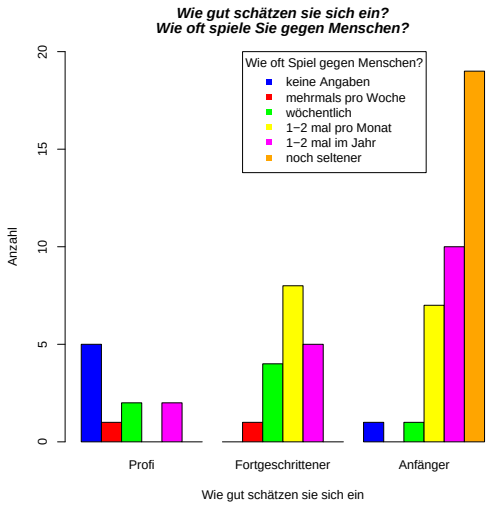



**Abbildung 31:** Wie gut hat man sich eingeschätzt?

Die Abbildung 31 stellt den Zusammenhang zwischen den Fragen „Wie gut schätzen Sie sich als Spieler ein“ und „Wie oft spielen Sie gegen Menschen“ dar. Dabei ist eine Tendenz erkennbar, dass diejenigen, die sich als *Anfänger* einstufen, eher seltener mit Menschen spielen. Umgekehrt sieht man auch, dass bei den *Profis* einige auch *mehrmals die Woche* oder *wöchentlich* angegeben haben. Diese Antworten waren verhältnismäßig stärker ausgeprägt als bei den *Anfängern*. Die *Fortgeschrittenen* tendieren eher zur Antwort *1-2 mal pro Monat*.
Diese Aussage lässt sich einfach erklären, da Anfänger eher weniger spielen als Profis oder Fortgeschrittene und dadurch auch seltener mit Menschen Poker spielen.

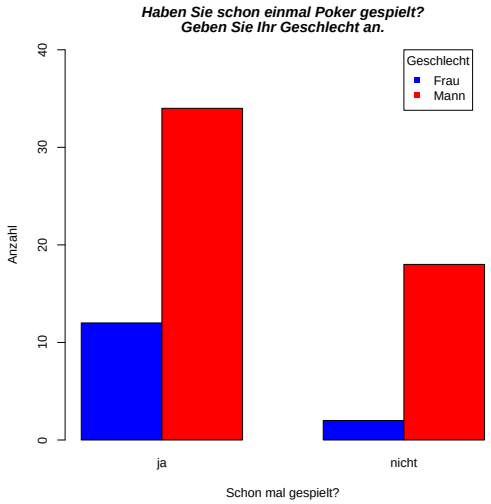



**Abbildung 32:** Hat der Befragte schon einmal Poker gespielt?

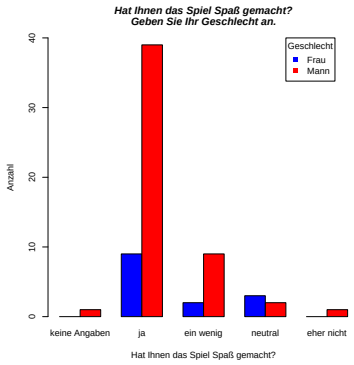



**Abbildung 33:** Hat den Befragten das Spielen Spaß gemacht?

Abbildung 32 und 33 zeigen ob der Befragte schon einmal Poker gespielt hat und ob unsere Implementierung des Spiels Spaß gemacht hat.
Es ist ersichtlich, dass die meisten Befragten schon einmal Poker gespielt haben und dass dem überwiegenden Anteil aller Befragten das mit Emotionen und Kommunikationsmöglichkeiten erweiterte PokerTH Spaß gemacht hat.
Die Aufschlüsselung nach Geschlechtern zeigt zusätzlich, dass der Großteil der Befragten aus Männern bestand (ca. doppelt so viele wie Frauen). Jedoch geht unabhängig vom Geschlecht die oben aufgestellten Kernaussagen aus den Diagrammen hervorgehen.

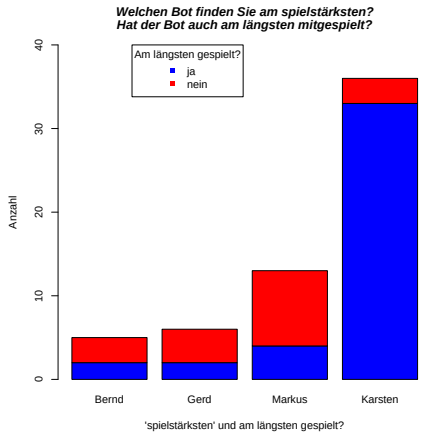



**Abbildung 34:** Welchen Bot finden Sie am spielstärksten? Hat der Bot auch am längsten mitgespielt?

Abbildung 34 beschreibt, welcher Gegenspieler von den Befragten als am spielstärksten angesehen wurde und unterteilt die Ergebnisse nach der Tatsache, ob mit dem ausgewählten Gegenspieler auch am längsten gespielt wurde.

Der Bot Karsten wird als am spielstärksten angesehen, vergleicht man die roten Balken, für die am spielstärksten angesehen Gegner, mit denen nicht am längsten gespielt wurde, so liegt hier Markus vor Gerd, Bernd und Karsten.

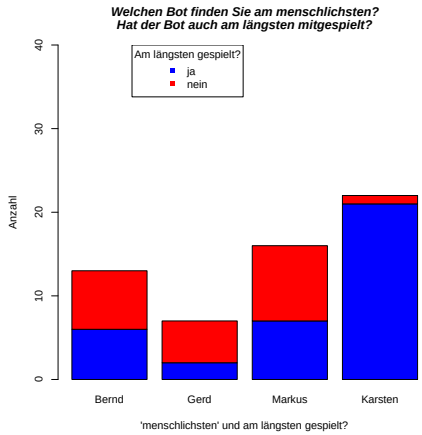



**Abbildung 35:** Welchen Bot finden Sie am Menschlichsten? Hat der Bot auch am längsten mitgespielt?

Das Diagramm 35 zeigt, welcher Gegenspieler von den Befragten als am menschlichsten galt. Unterteilt ist das Diagramm weiterhin danach, ob auch gegen den ausgewählten Gegner am längsten gespielt wurde.

Insgesamt liegt der PokerTH-Bot Karsten leicht in Führung vor Markus, gefolgt von Bernd und zuletzt Gerd. Es ist jedoch sehr eindeutig zu sehen, dass ein von der Projektgruppe erstellten Gegenspieler von denjenigen Befragten als deutlich menschlicher angesehen wurden, auch wenn gegen ihn nicht am längsten gespielt wurde. Die Eigenschaft menschlich wurde von diesen Befragten also nicht mit möglichst lange am Tisch bleiben gleichgesetzt.

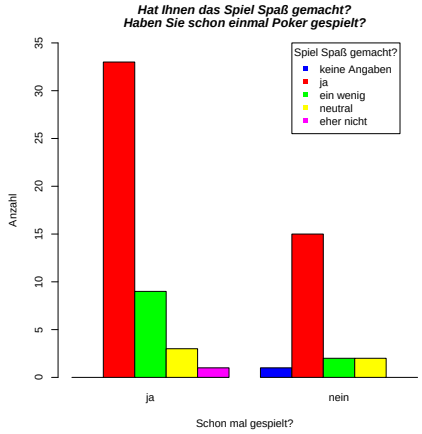



**Abbildung 36:** Hat Ihnen das Spiel Spaß gemacht? Haben Sie schon einmal Poker gespielt?

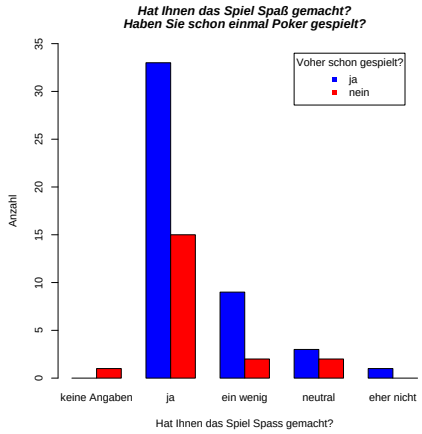



**Abbildung 37:** Hat Ihnen das Spiel Spaß gemacht? Haben Sie schon einmal Poker gespielt?

Die Balkendiagramme 36, und 37 beschreiben, ob den Befragten das Spiel Spaß gemacht hat und unterteilt die Antworten danach, ob die Probanden überhaupt schon einmal Poker gespielt haben.

Es ist eindeutig, dass sowohl den Befragten das Spiel Spaß gemacht hat, die schon einmal Poker gespielt haben, als auch denjenigen, dies es bisher noch nicht gespielt haben.

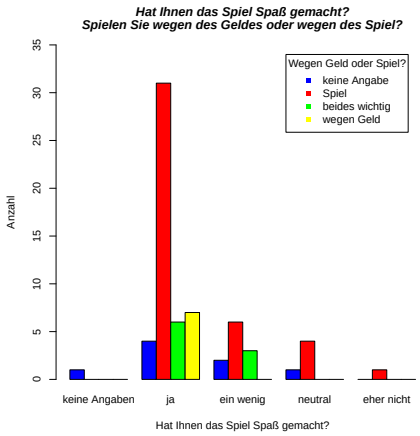



**Abbildung 38:** Hat Ihnen das Spiel Spaß gemacht? Spielen Sie wegen des Geldes oder wegen des Spiels?

Die Abbildung 38 stellt dar, ob einem Befragten das Spiel Spaß gemacht hat und unterteilt weiter danach, ob der Befragte spielt um Geld zu gewinnen, oder wegen des Spiels.

Man erkennt, dass sehr vielen Befragten das Spiel Spaß gemacht hat und dass der Großteil aus dieser Gruppe Poker nicht wegen des Geldes spielen.

### 3.9.3 Korrelationsanalyse

Eine Korrelation zwischen zwei Variablen sagt aus, dass mit steigendem Wert der ersten Variable $A$ auch eine Steigerung der zweiten Variable $B$ einhergeht. Ein negativer Korrelationskoeffizient sagt aus, dass mit steigendem $A$ die Variable $B$ sinkt. Berechnet wird der Wert mit der Formel

$$\varrho(A,B) = \frac{E\left((A - E(A)) \cdot (B - E(B))\right)}{\sqrt{Var(A)} \cdot \sqrt{Var(B)}}$$

wobei $E()$ den Erwartungswert und $Var()$ die Varianz einer Ausprägung angeben. Aus dem Aufbau der Formel wird auch ersichtlich, dass diese Relation symmetrisch ist, d.h. wenn $A$ steigt, steigt auch $B$ und umgekehrt.

Beispielsweise hat die Variable „Größe eines Menschen“ mit „Gewicht eines Menschen“ einen positiven Korrelationskoeffizienten, da es allgemein beobachtet werden kann, dass ein größerer Mensch meistens auch mehr wiegt. Dieser direkte kausale Zusammenhang (Masse nimmt Volumen ein) ist nicht immer gegeben, und somit ist eine zusätzliche Interpretation des Koeffizienten nötig.

Der Koeffizient hat dabei den Wertebereich $[-1;1]$. Sowohl $-1$ als auch 1 geben die strengste Form einer Relation zwischen den Variablen an, da eine Steigerung der einen Variablen dann immer auch eine Steigerung (oder bei $-1$ auch Senkung) der anderen Variablen mit sich führt. Je weiter sich der Wert der 0 annähert, desto schwächer ist der oben beschriebene Zusammenhang, wobei er bei 0 nicht mehr gegeben ist.

Um auffällige Zusammenhänge zwischen den Variablen des Fragebogens zu ermitteln, wurde mit der OpenSource-Statistiksoftware „R“ eine Korrelationskoeffizientenmatrix aufgestellt. Diese beinhaltet alle Korrelationskoeffizienten zwischen allen Paarungen von Variablen. Da die Antwortmöglichkeiten unserer Fragen immer eine Ordnung haben, ist

das Errechnen der Korrelationskoeffizientenmatrix sinnvoll. Da betragsmäßig geringe Korrelationen wenig Aussagekraft haben, wurden nur alle Werte $> 0.4$ betrachtet.

Die Korrelation zwischen den Fragen „Wie alt“ und „Bildungsstand“ ist ausreichend hoch, was den Erwartungen entspricht, da die Umfrage im universitätsnahem Raum stattfand und viele Versuchspersonen einer ähnlichen Altersgruppe abgedeckt hat.

Ebenso ist $\varrho$(„Wie oft spielen Sie gegen Menschen?“, „Wie gut schätzen Sie sich als Pokerspieler ein“) mit $\approx 0.675$ hoch, da Personen, die häufiger gegen andere Personen spielen auch besser Poker spielen können und sich damit auch besser einschätzen (vgl. Abb. 31).

Eine für die Projektarbeit befriedigender Wert findet sich bei dem Variablenpaar („Fanden Sie die Kommunikation sinnvoll?“, „Hat Ihnen das Spiel Spaß gemacht?“). Mit einem Koeffizienten von 0.446 korrelieren die Variablen miteinander, weswegen man vermuten kann, dass die zusätzlich eingebauten Features, welche die Kommunikation ermöglichen, zum Spielspaß der Probanden beigetragen haben.

Der erhöhte Wert zwischen „Wie gut fanden Sie den Einsatz der Emoticons?“ und „Hat Ihnen das Spiel Spaß gemacht?“ lässt ebenfalls vermuten, dass das Konzept der Emoticons eine Bereicherung für das Spiel darstellen.

Ein weiterer Koeffizient von 0.422 lässt vermuten, dass die Probanten, die den Einsatz der Emoticons gut bewertet haben, tendenziell auch mehr Nachrichten verschickt haben.

### 3.9.4 Assoziazionsregeln

Eine weitere Technik der Datenanalyse kommt aus dem Bereich des *Data Minings*: Assoziazionsregeln lernen.
Ziel hierbei ist es, unter „häufig“ vorkommenden Mengen Regeln zu ermitteln, aus denen abzulesen ist, welche Mengen weitere Mengen nach sich ziehen. Detaillierter wird auf das Thema in Kapitel „Assoziationsregeln“ des Seminars „Maschinelles Lernen/regelbasierte Steuerung“ (Abschnitt G.2.1) eingegangen.

Der Apriori-Algorithmus liefert alle häufigen Mengen, welche einen gewissen Mindestsupport erfüllen und sucht mit diesen nach Regeln, welche eine geforderte Mindestkonfidenz erfüllen. Es wurde für die Suche nach Assoziationsregeln die Statistiksoftware „R“ verwendet, in der der genannte Algorithmus implementiert ist.

Die resultierenden Regeln müssen jedoch manuel analysiert werden, da viele offensichtliche oder auch redundante Regeln gebildet werden. So brachte die Assoziationsregel (Geschlecht = Mann) → (Wie gut schätzen sie sich ein = Anfänger) keine neuen Erkenntnisse, da der Großteil der Befragten Männer waren, die angaben, dass sie Anfänger sind. Aus dem Grund wurden die Regeln, welche männliche Befragten in Verbindung brachten, nicht weiter beachtet.

Für die folgenden Regeln wurde ein Mindestsupport von $0,4$ und eine Mindestkonfidenz von $0,5$ festgelegt:

| Nr | Regel | Support | Konfidenz |
|---|---|---|---|
| 1 | (Passiv/Agressiv = teils teils) →(Spaß gemacht = ja) | 0.4242 | 0.8485 |
| 2 | (Spaß gemacht =ja) →(Passiv/Agressiv = teils teils) | 0.4242 | 0.5833 |
| 3 | (Geld/Spiel = Spiel) →(Schon einmal gespielt = nein) | 0.5000 | 0.7857 |
| 4 | (Schon einmal gespielt = nein)→(Geld/Spiel = Spiel) | 0.5000 | 0.7174 |
| 5 | (Schon einmal gespielt = nein)→(Spaß gemacht = ja) | 0.5000 | 0.7174 |
| 6 | (Spaß gemacht = ja) →(Schon einmal gespielt = nein) | 0.5000 | 0.6875 |
| 7 | (Geld/Spiel = Spiel) →(Spaß gemacht = ja) | 0.4697 | 0.7381 |
| 8 | (Spaß gemacht = ja) →(Geld/Spiel = Spiel) | 0.4697 | 0.6458 |

Die Regeln 5 und 6 werden in den Abbildungen 36 und 37 verdeutlicht, die Regeln 7 und 8 in Abbildung 38.

# 4 Erfahrungen

## 4.1 Einleitung

Da es sich für die meisten Teilnehmer der Gruppe um das erste größere IT-Projekt handelte, kam es erwartungsgemäß zu einer Reihe von Problemen, mit der die Gruppe konfrontiert wurde. Auf der einen Seite hatten wir Probleme im Bereich der Kommunikation, wie etwa Planung und Verteilung von Aufgaben, und auf der anderen Seite Schwierigkeiten im technischen Bereich. Im Folgenden wird zunächst auf die Probleme aus dem Bereich der Kommunikation eingegangen und danach auf die technischen. Abschliessend werden die Ergebnisse in einem kurzen Fazit zusammengefasst.

## 4.2 Organisatorische Probleme

Im Bereich der Organisation werden folgende Punkten näher beschrieben:

- Planung des Projekt
- Deadlines
- Kommunikation
- Die gelbe Karte
- Der Projektleiter

In den nächsten Unterkapiteln wird versucht, die Problematiken für Aussenstehende verständlich zu machen und die Vorgehensweise der Gruppe zu beschreiben.

### 4.2.1 Planung des Projekt

Als wir uns für die Umsetzung einer KI für Poker entschieden hatten, wurde schnell klar, dass die Vorstellungen bezüglich Zielvorgaben und Realisierungsansätzen zwischen den verschiedenen PG-Mitgliedern sehr weit auseinander gingen. Manche PGler gingen von dem Ansatz aus einen perfekten Pokerspieler zu entwickeln um diesen dann künstlich abzuschwächen wohingegen andere nur eine Schnittstelle entwerfen wollten um ein Spiel um Emotionen erweitern zu können. Am Anfang musste zunächst entschieden werden, ob man ein komplett neues Poker programmiert, auf eine rudimentäre Pokerplattform aufbaut oder ein bereits fertiges Spiel erweitert. Außerdem gab es eine große Debatte zum Thema, ob man lieber Java nutzen soll, da diese Sprache sehr portabel ist, oder ob man lieber mit C++ entwickelt, um eventuelle Geschwindigkeitsvorteile erzielen zu können. Die Diskussionen am Anfang waren mit Sicherheit notwendig, jedoch hätte man durch eine kompaktere Diskussion Zeit gewinnen können, welche man am Ende des Semesters gebraucht hätte.

### 4.2.2 Deadlines

Die Gruppe hatte sich selber Milestones definiert, die zu bestimmten Daten erreicht werden sollten. Dies hatte den Zweck mehr Struktur in die Arbeit zu bringen und den Fortschritt des Projektes besser messbar zu machen. Insgesamt gab es vier Milestones (vgl. Abschnitt 3.4 auf Seite 14). Leider konnte keiner der Milestones rechtzeitig erreicht

werden. Dies lag zum einen daran, dass die Milestones sehr optimistisch festgelegt wurden, und daran, dass unerwartete Probleme auftraten. So brauchte z.B. das KI-Modul durch zahlreiche Veränderungen viel länger als erwartet und die Einbindung der KI in die PokerTH-Umgebung war ebenfalls schwerer realisierbar als geplant.

### 4.2.3 Kommunikation

Die Kommunikation in der Gruppe beschränkte sich in der ersten Zeit auf die Gespräche in den PG-Sitzungen. Da für die Entwicklung die Gruppe in drei Arbeitsgruppen aufgeteilt wurde, wurde die Kommunikation zudem noch erschwert. Die den Gruppen zugeteilten Aufgaben wurden intern bearbeitet und einer aus jeder Gruppe stellte dann in der Sitzung die Ergebnisse vor. Zwar wurde in einer der Sitzungen vorgeschlagen für jede der Arbeitsgruppen einen Sprecher zu wählen, dieser Vorschlag wurde allerdings abgelehnt. Diese unkoordinierte Kommunikation hatte diverse Nachteile:

- Die Definition der Schnittstellen zwischen den einzelnen Gruppen gestaltete sich als problematisch.
- Bei Problemen mit Modulen einer anderen Gruppe hatte man keinen direkten Ansprechpartner.
- Teilweise zerfielen die Arbeitsgruppen wiederum in Subgruppen, so dass selbst gruppenintern nicht immer klar war welche Aufgaben bearbeitet wurden.
- Die einzelnen Arbeitsgruppen waren so sehr mit ihren Teilaufgaben beschäftigt, dass zu oft der Blick für das Ganze verloren ging.

### 4.2.4 Die gelbe Karte

Am 19. Juni 2008 erhielten die PGler von den Betreuern eine E-Mail, in welcher diese die Befürchtung äußerten, dass das PokerTH Projekt durch die Projektgruppe nicht erfolgreich zu Ende geführt werden könnte. Insbesondere kritisierten sie folgende Aspekte:

- Die PG scheitert an technischen Problemen wie etwa Speicherallokationsfehlern, weshalb das KI-Modul viel zu weit in der Entwicklung zurückhängt.
- Die Stimmung in der Gruppe ist schlecht und es gibt Konflikte zwischen den Gruppenmitgliedern.
- Durch wiederkehrendes verschieben der Deadlines wird zu viel der verbleibenden Zeit für Handwerk aufgebraucht statt sich mit dem eigentlichen Thema der emotionalen und menschähnlichen Spielweise der Bots beschäftigen zu können.

Um das Projekt in der verbleibenden Zeit in zufriedenstellende Bahnen zu lenken, wollen die Betreuer sich intensiver mit der Führung der Gruppe beschäftigen.

### 4.2.5 Der Projektleiter

Eine wesentliche Neuerung, die nach der gelben Karte durch die Betreuer eingeführt wurde, war die Einführung eines Projektleiters. Der als Projektleiter gewählte Marc hatte die Aufgabe die Aufgabenverteilung zu koordinieren, den Überblick über den Fortschritt des gesamten Projekts zu haben, sowie problemspezifische Anfragen an die zuständigen Personen weiterzuleiten. Außerdem sollte er sich um die vermehrte Nutzung des TRAC Ticketsystems kümmern, welches vorher nur sehr wenig genutzt worden war. Durch die Einführung des Projektleiters und durch die Verwarnung in Form der gelben Karte wurden direkt Verbesserungen sichtbar.

- Durch die Kontrollinstanz des Projektleiters und die massive Kritik durch die Betreuer wurden langwierige Programmier-Baustellen fertiggestellt.
- Da der Projektleiter in ständiger Absprache mit den einzelnen PGlern war, wusste er stets wo Hilfe benötigt wurde und konnte Personalmangel schnell ausgleichen.
- Es kam zu einem drastischen Fortschritt in der Entwicklung, so dass sich die Stimmung in der Gruppe sehr verbesserte.

## 4.3 Technische Probleme

Auf der Seite der technischen Probleme werden folgende Punkte angesprochen:

- Probleme mit der Programmiersprache
- Arbeiten an der Uni
- Der Umgang mit dem Trunk

### 4.3.1 Probleme mit der Programmiersprache

Ein Großteil der PGler hatte sich im Verlauf des Studiums hauptsächlich mit der Programmiersprache Java beschäftigt. Nur Wenige hatten bereits vor dem Beginn der Projektgruppe Erfahrungen mit C++ sammeln können. Zwar gab es am Anfang des Semesters Tutorials zu den Themen C++ und der Übersetzung von PokerTH, jedoch kam es immer wieder im Verlauf des Projektes zu Schwierigkeiten bei der Implementierung und Übersetzung. So gab es z.B. zahlreiche Segmentation Faults, Linkererror sowie Probleme mit den makefiles und qmake. Da sich Marc im Verlauf des Semesters intensiv mit dem Compiler auseinander setzte, gelang es immer schneller die Übersetzungsprobleme in den Griff zu bekommen.

### 4.3.2 Arbeiten an der Uni

Ein Teil der Gruppe entwickelte hauptsächlich in der Uni an unserem Programm weiter. Leider gab es mit den zur Verfügung gestellten Rechner einige Schwierigkeiten. Hierzu zählten:

- Die boost-library, welche für die Entwicklung benötigt wurde, war nur auf wenigen speziellen Rechnern vorhanden.
- Die Nutzung des SVN funktionierte über lange Zeit nicht auf den zur Verfügung gestellten Rechnern.
- Der Zugriff auf das TRAC von den Universitätsrechnern wurde erst nach der Einrichtung eines Proxyservers durch Marc möglich.
- Die meisten PGler waren mit der Benutzung des Batch-Systems nicht vertraut und wussten nicht wie und für welche Aufgaben man dieses gut nutzen kann.

### 4.3.3 Der Umgang mit dem Trunk

Leider kam es einige Male vor, dass die Arbeit durch Probleme mit dem Trunk verzögert wurden. Folgende Probleme traten auf:

- Der Sourcecode, der commitet wurde, wurde nicht hinreichend getestet.
- Es wurden unterschiedliche Compilerversionen benutzt.

- Die Compilerflags waren unterschiedlich gesetzt, so dass manche Situationen bei Einigen eine Warnung hervorriefen und bei Anderen einen Fehler produzierten.

## 4.4 Fazit

Während des ersten Semesters war es vor allem die mangelnde Kommunikation, welche zu teils erheblichen Verspätungen führte. Organisatorische Aspekte wie etwa Entscheidungen über das Spiel und die Programmiersprache hätten genau wie ein Einrichten eines TRAC und SVN im Vorfeld, eine Menge Zeit einsparen können, um so mehr Zeit für die eigentlichen Aufgaben zu haben. Die Wahl des Projektleiters hat dazu geführt, dass die Aufgaben konsequenter ausgeführt wurden und der Informationsfluss signifikant verbessert wurde. Aus diesem Grund ist es höchstwahrscheinlich sinnvoll auch im zweiten Projekt wieder einen Projektleiter einzusetzen. Dadurch, dass sich die PG Mitglieder während der Implementierungsphase intensiv mit C++ auseinandergesetzt haben, sollten bei der Realisierung des zweiten Projekts wesentlich weniger Schwierigkeiten auftreten als im ersten Projekt.

# 5 Zusammenfassung

Die Projektgruppe hatte die Aufgabe, CI-Methoden auf ein Spiel anzuwenden, um menschlicheres Verhalten der Computergegner zu erzielen und somit den Spielspaß zu erhöhen. Um hierfür ein gemeinsames Wissensfundament zu schaffen, wurde innerhalb der Seminarphase nötiges Grundwissen über CI-Methoden wie z.B. Neuronale Netze und Evolutionäre Algorithmen aber auch fächerübergreifendes Wissen aus den Bereichen Soziologie und Psychologie vermittelt, da das Erstellen eines Emotionsmodells einen wichtigen Aspekt in unserem Projekt darstellen sollte. Um die Entscheidungsfindung für ein Spiel zu erleichtern und den aktuellen Forschungsstand der drei zur Auswahl stehenden Spiele Poker, Diplomacy und Junta darzulegen, wurde zu diesem Thema ebenfalls ein Seminarvortrag gehalten. Die Gruppe entschied sich nach Abwägen der Vor- und Nachteile der einzelnen Spiele für Poker. Eine große Herausforderung war es, menschliches Verhalten und die Fähigkeit geeignete Kommunikationskanäle für Emotionen in einem Spiel zu realisieren, welches darauf ausgelegt ist die Emotionen nach Möglichkeit zu verbergen.

Während sich ein Teil der Gruppe mit der Implementierung der CI-Methoden beschäftigte, entwarf der andere Teil der Gruppe ein Charaktermodell, welches in der Lage sein sollte unterschiedliche Spielertypen widerspiegeln zu können und somit menschenähnlicheres Verhalten zu ermöglichen. Die Charaktere wurden zudem durch das Hinzufügen eines Gedächtnisses abgerundet, welches in der Lage sein sollte, Spielinformationen über die einzelnen Spieler zu sammeln, um diese besser einschätzen zu können und die eigene Spielweise zu optimieren. Man entschied sich dafür, das Opensource-Spiel PokerTH dahingehend zu modifizieren die integrierte Nachrichtenkommunikation auch für CI-Player zu ermöglichen und als weiteren Kommunikationskanal eine Anzeigemöglichkeit von Emoticons hinzuzufügen.

Die für die Implementierung einer künstlichen Intelligenz zugehörigen Gruppe entwarf ein Neuronales Netz, welches spielbezogene Eingabeparameter auf das Spielverhalten während der Pokerrunde abbildete. Es wurden zwei unterschiedliche CI-Lernmethoden implementiert, wobei die eine auf einem Backpropagationansatz beruhte und die andere auf einem evolutionären Ansatz. Ein Vergleich auf Testdaten zeigte, dass sich der evolutionäre Ansatz für unsere Problematik als besser geeignet erwies.

Die nach internen Tests durchgeführte Umfrage unter Studenten auf dem Campus zeigte, dass unser Ziel, einen menschenähnlicheren Computerspieler als den verglichenen statischen PokerTH-Bot zu kreieren, erfolgreich war. Laut Umfrage waren von Zufall stark abweichende Unterschiede in der Spielweise der einzelnen CI-Spieler festzustellen. Auch wurde durch die Einführung der Emoticons der Spielspaß vergrößert.

# 6 Ausblick

Nachdem in den vorherigen Kapiteln bereits ein abschliessendes Resümee des Projektverlaufs des ersten Semesters gezogen worden ist, möchten wir nun einen kleinen Ausblick auf das zweite Semester dieser Projektgruppe wagen.

Für das zweite Projekt wird unabhängig von den eingesetzten CI-Methoden die Konstruktion einer menschenähnlichen Künstlichen Intelligenz sehr viel komplexer sein. Der kontinuierliche Spielverlauf eines Echtzeit-Strategiespiels mit „gleichzeitigen“ Ereignissen an vielen unterschiedlichen Punkten des Spielfeldes werden es uns schwer machen, einen Gegenspieler zu konstruieren, der über die gesamte Dauer des Spiels einen menschenähnlichen Eindruck macht. Die Komplexität der Aufgabenstellung für das zweite Projekt hängt mit der Komplexität des Spiels zusammen, so dass bei der Auswahl eines anspruchsvollen Spiels die Erwartungshaltung für die umzusetzenden Ziele verringert werden muss. Andererseits muss die Zielsetzung höher gesteckt werden, wenn das Spiel weniger anspruchsvoll ist. Es muss unsere Intention sein, einen menschenlichen NPC zu modellieren, der über das gesamte Spiel ein konsistentes Verhalten zeigt. Jedoch kann davon ausgegangen werden, dass wir uns auf einige zentrale Aspekte konzentrieren müssen, so dass der Computergegener nur in einigen Spielsituationen menschlich wirken kann, allerdings nicht in jeder Situation. Die Vielzahl unterschiedlicher Einheiten und die gegenseitigen Wechselbeziehungen untereinander erschweren unsere Entwicklung zusätzlich, obgleich genau diese Masse an Handlungssträngen für Entwickler und Spieler äusserst interessant ist. Es wäre sinnvoll zu erörtern, inwiefern wir die von uns eingesetzten Emotionen bzw. ein Botgedächtnis in einem Echtzeit-Strategiespiel einbauen können. Gleichwohl ist die Projektgruppe der Auffassung, dass wir unseren Blickwinkel unter Umständen verschieben müssen, um uns nicht frühzeitig darauf festzulegen, dass der Einsatz von Emotionen bzw. einem Gedächtnis zum gewünschten Ziel führt. Wie bereits erwähnt wird als Spielgenre im zweiten Projekt wahrscheinlich das Genre der Strategiespiele als Ausgangspunkt unserer Arbeit stehen, jedoch ist nicht ausgeschlossen, dass sich die Projektgruppe einem anderen Genre bedient. Wir sind uns im Klaren, dass sich viele Verbesserungen einbauen lassen. So ist die Anordnung und Auswahl der Gebäude des NPCs nach nur wenigen Stunden Einspielzeit leicht vorherzusagen, so dass der Spielspass unter dem Spielverhalten leidet. Hier muss eine Strategie entwickelt werden, die die Anordnung und Auswahl an Gebäuden sinnvoll verbessert. Es ist denkbar, dass gleiche Gebäudetypen nebeneinander platziert werden, um eine kompaktere Bauweise zu ermöglichen. Des weiteren können Abwehrtürme, die gegenerische Einheiten bekämpfen sollen, ebenfalls nebeneinander gesetzt werden, so dass gegnerische Einheiten nicht wie zuvor von einem Abwehrturm, sondern nun von mehreren Abwehrtürmen angegeriffen werden können. Einheiten sollen sich angemessen verhalten, das heißt, sie sollen als Kollektiv interagieren und sich dem Verhalten des Spieles angemessen anpassen. Dies ist nur zu erreichen, wenn Einheiten untereinander in Interaktion stehen oder von einer höheren Instanz koordiniert werden. Ausserdem wäre es wünschenswert, wenn sich Einheiten in bestimmten Situationen zu Gruppen angemessener Größe formieren, damit ein bestimmtes Ziel erreicht wird.

Für die Koordination und Planung des zweiten Projekts ist bislang lediglich eine Diskussion angestoßen worden, welchem Spiel die Projektgruppe sich im zweiten Semester widmen wird. Hier bleiben noch viele Fragen offen, die noch geklärt werden müssen. So hat es sich als weniger sinnvoll erwiesen, die Konstruktion des Neuronalen Netzes einer bestimmten Gruppe zu überlassen. Die Fertigstellung dieses Moduls hat den gesamten Projektverlauf nachträglich verzögert, woraufhin im zweiten Projekt sicherlich die Konzentration auf die Entwicklung einer geeignten Künstlichen Intelligenz gelegt werden muss. Eine generelle Einteilung in Gruppen ist unserer Meinung nach obligatorisch. Die

Projektgruppe muss sich nur darauf einigen, welche Konzepte umgesetzt werden sollen. So kann die Anordnung und Wahl von Gebäuden oder die Gruppierung und Bewegung von Einheiten ein mögliches Ziel sein, dass implementiert werden soll. Bei dem Einsatz geeigneter CI-Methoden kann auf bisher gewonnene Erkenntnisse zurückgegriffen werden. So wurde ein Neuronales Netz, das anhand von Eingabemustern, die während eines Spiels abgespeichert wurden, unter Einsatzes eines Evolutionären Algorithmus trainiert. Inwiefern dieser Ansatz bei einem Echtzeit-Strategiespiel umzusetzen ist, wird uns beschäftigen, aber aufgrund der hohen Informationsmenge wird evtl. ein anderer Ansatz gewählt.

Bei der Konzeption der Künstlichen Intelligenz hat sich die Projektgruppe ausschliesslich auf Neuronale Netze und Evolutionäre Algorithmen konzentriert. Denkbar wäre eine weitere interessante Methode im Bereich der Computational Intelligence, nämlich die Fuzzy-Logik. Die Fuzzy-Logik beschäftigt sich mit unscharfer Logik, mit deren Hilfe es möglich ist, aus einer verbalen Beschreibung Regelsysteme zu produzieren. Auch die Kombination mehrerer CI-Techniken ist denkbar.

# A Avatare und Kommunikation mit Computerspielern

## A.1 Vorwort

Diese Ausarbeitung entstand im Rahmen der Projektgruppe 529, die zum Thema die Modellierung menschenähnlicher Gegenspieler in Strategiespielen mit Techniken der Computational Intelligence hat. Die Ausarbeitung beschäftigt sich mit Avataren und der Kommunikation in Computerspielen.

Bezüglich der Avatare gehe ich darauf ein, welche Funktion sie in Spielen einnehmen und vor allem, welche Rolle sie für den Spieler spielen. Hierbei gehe ich neben dem, was man mit Avataren ausdrücken kann, besonders auf den psychologischen Aspekt ein. Dies geschieht am Beispiel von "The Palace", einer virtuellen Umgebung die auch als "Life Sim" bezeichnet wird. Nun gibt es gegen die Verwendung von "The Palace" als Beispiel berechtige Einwende. Zum Einen ist es nicht auf dem aktuellen Stand der Technik, doch die psychologischen Aspekte der Avatare sind die selben wie bei heute aktuellen Umgebungen, zum Beispiel "Second Life" (vgl. [Sul07], Lookin' Good, or Not: The Avatar). Zum anderen ist es kein Computerspiel im herkömmlichen Sinne. Der signifikante Unterschied zwischen einer solchen virtuellen Umgebung, die man auch als einen virtuellen Chat bezeichnen könnte, ist, dass es kein vorgegebenes Spielziel gibt. Doch wie es bei Rollenspielern nicht unüblich ist, kann man auch das Spielen einer Rolle als Spielsinn ansehen. Dies kann man in "The Palace" eben so gut wie in MMORPGs[1], von denen "World of Warcraft" und "The Dark Age of Camelot" bekannte Vertreter sind. Deswegen neige ich dazu, auch "The Palace" als Spiel anzusehen.

Hinsichtlich der Kommunikation betrachte ich verschiedenen Möglichkeiten, wie Spieler mit einander interagieren können, welche Funktionen die Kommunikation hat und worüber sich Spieler unterhalten. Neben diesen Schwerpunkten schneide ich kurz die Themen an, wie Chatbots versuchen, menschlich zu wirken, und wie man die Textgenerierung in Spielen individueller gestalten und an das Spielgeschehen und den Spieler anpassen kann.

## A.2 Avatare

### A.2.1 Was ist ein Avatar

Avatara bezeichnet im hinduistischen Glauben die menschliche Verkörperung eine Gottes oder einer göttlichen Eigenschaft und bedeutet "Abstieg".
In der hiesigen Gesellschaft ist die Definition des Avatars als ein visueller Stellvertreter in einer virtuellen Welt gebräuchlicher. Avatare finden sich in Internetforen, Chats und Computerspielen wieder. In Foren kann man in der Regel bei seinem Benutzernamen ein kleines Bild anzeigen lassen, was Avatar genannt wird. In Computerspielen oder im virtuellen Welten wie "Second Life" sind Avatare 3D-Spielfiguren, die vom Benutzer gesteuert werden. Allgemein gibt es keine Beschränkungen, wie ein Avatar aussieht oder was er darstellt.

[1] Massive(ly) Multiplayer Online Role-Playing Game, ein Computerrollenspiel für eine große Anzahl an Mitspielern, welches über das Internet gespielt wird.

### A.2.2 Ausdruckskraft

Nach Tony Manninen und Tomi Kujanpää (vgl. [MK05]) hat das Aussehen eines Avatars im Spiel "Battlefield 1942" zwei Funktionen. Zum Einen soll die Teamzugehörigkeit visualisiert werden. Zum Zweiten lässt sich die Rolle, die der Spieler übernimmt, erkennen. Anhand der Uniform und der Waffe kann man unterscheiden, ob es sich um einen Scharfschützen oder einen Angehörigen eines Anti-Panzer-Trupps handelt. Ein Rotes Kreuz gibt unmissverständlich zu verstehen, dass er einen Sanitäter spielt. So lassen sich auch Schlüsse ziehen, wie der Spieler sich verhalten wird und, für die Gegner, wie gefährlich er in einer Situation für sie ist. Weiter erwähnen die Autoren, dass durch Bewegung eines Avatars die aktuelle Aktion eines Spielers verdeutlicht wird (schießt er, zielt er, wirf er eine Granate oder wird er getroffen).
Die von Tony Manninen und Tomi Kujanpää bezüglich "Battlefield 1942" gemachten Aussagen lassen sich natürlich auch auf andere Spiele übertragen. Durch den Avatar und sichtbare Ausrüstung kann gezeigt werden, welche Rolle er spielt und welche Fähigkeiten er hat. Ein Magier in einem Rollenspiel kann an seiner Robe und ein Krieger an Waffen und Rüstung erkannt werden. Durch die Ausrüstung kann neben Fähigkeiten, eine Rüstung verringert Schaden, auch die Mächtigkeit, zum Beispiel durch ein besonderes Schwert, abgelesen werden (vgl. [Man03], Abschnitt 4.3 und 4.4). So können auch erfahrene von unerfahrenen Spielern unterschieden werden.
Neben den oben genannten Informationen, die Avatare vermitteln, kann mit ihnen auch explizit kommuniziert werden (siehe Kapitel A.3.1) und sie vermitteln Informationen über die Person, die einen Avatar spielt (siehe Kapitel A.2.5).

### A.2.3 Verwendung in Strategiespielen

In Strategiespielen finden Avatare häufig keine Verwendung. Dies könnte daran liegen, dass der Spieler zwar das Spielgeschehen bestimmt indem er Einheiten befehligt und Gebäude baut, jedoch nicht selber in diesem direkt mitwirkt. In der Regel werden Einheiten aus einer nicht fokussierten Vogelperspektive gesteuert, so dass der Spieler alles beobachten kann. Er steht also nicht selber auf dem Schlachtfeld, wenn er Befehle erteilt.
Es gibt allerdings Spiele, bei denen versucht wird, den Spieler stärker mit einer Einheit zu verbinden. Das Spiel "Total Annihilation" verwendet eine besondere Einheit, den sogenannten "Commander". Dieser ist zwar auch keine direkte Online-Persönlichkeit des Spielers, jedoch eine sehr mächtige Einheit, die eine zentrale Rolle spielt. Mit ihr beginnt das Spiel und durch ihre Zerstörung wird das Spiel beendet. So ist sie für den Spieler sehr wichtig und bleibt ihm nachhaltig in Erinnerung.
Einen Schritt weiter geht das Echtzeit-Strategie Spiel "Sacrifice". Hier übernimmt der Spieler die Rolle eines Zauberers, welchen er in der Third Person Perspektive[2] steuert. Er kann seine Einheiten aus sicherer Entfernung befehligen oder selber direkt am Kampf teilnehmen. Natürlich ist es hier schwieriger die Übersicht zu behalten, als aus der Vogelperspektive, weil man nur das sieht, was sich im Blickbereich des Avatars befindet.
Eine weitere Möglichkeit, einen Avatar in Strategiespielen zum Einsatz zu bringen, wäre das anzeigen eines Avatars in einer Leiste am Rande des Bildschirms. Aus Platzgründen würde es sich entweder auf ein kleines individuelles Bild, welches der Benutzer frei wählen kann (ähnlich wie in Internet Foren), oder auf einen Kopf beschränken. Den Kopf könnte der Spieler seinen eigenen Vorstellungen anpassen. In dieser Leiste könnten alle Avatare der sich am Spiel beteiligenden Spieler angezeigt werden. Ihre Gesichtsmimik könnte durch die Benutzer selber gesteuert werden oder sich automatisch an das Spielgeschehen anpassen. Durch einen grimmigen Gesichtsausdruck würde man erkenne, das ein Spieler gerade angegriffen wird oder sein Angriff keinen Erfolg hatte. Welche

[2] Die Kamera befindet sich immer im Rücken des Avatars und ist auf diesen fokussiert.

Avatare in der Leiste angezeigt werden, könnte auch weiter beschränkt sein, da sonst Rückschlüsse auf Spielhandlungen möglich sind, die man selber nicht sehen kann. Es könnten nur die Avatare der verbündeten Spieler angezeigt werden. Oder es muss jeder Spieler seine Zustimmung geben, wenn sein Avatar bei einem anderen angezeigt werden soll.
Um eine ganze Figur und die Umgebung, in der der Spieler sich befinden könnte (ein Planungsraum in einem Bunker oder ein Gefechtsstand im Feld), mit einzubeziehen, könnte eine Art Videobotschaft benutzt werden. Ähnliches wird bei manchen Spielen, darunter "Command & Conquer", zum Briefing eingesetzt. Jeder Spieler könnte vorgefertigte Videobotschaften an andere Spieler schicken. In ihnen würde dann sein Avatar in einem Bunker zu sehen sein, wie er dem Nachrichtenempfänger auf unfreundliche Weise mitteilt, dass er seine Truppen abziehen soll, oder wie er ihn für einen missglückten Angriff auslacht.

### A.2.4 Nur eine Spielfigur?

Computerspiele bieten die Möglichkeit, in andere Welt einzutauchen und in diesen Welten Rollen zu spielen. Weder die Welten noch die gespielten Rollen müssen viel mit der Realität zu tun haben. Dies bedeutet aber auch, dass man Dinge ausprobieren kann, die man schon immer einmal ausprobieren wollte. So kann jemand, der in der Realität introvertiert ist, im Spiel als Kämpfer für seine Ideale einstehen. Dieses Verhalten ist etwas, was er sich im realen Leben nie trauen würde. Nicholas Yee hat bei seinen Umfragen bei Spielern von fünf MMORPGs[1] (vgl. [Yee02]) festgestellt, dass ca. 25% der Befragten im Spiel mehr sie selbst sind als im realen Leben. Seine Aussage diesbezüglich bezieht sich auf das Verhalten der Spieler im Spiel. Da die Charaktererstellung und somit die Avatargestaltung ein Teil des Spieles ist und in ihr viel Ausdruckskraft liegt, ist es auch nicht überraschend, dass man diese Aussage auf den Avatar als Spielfigur übertragen kann. So kann jemand, der bestimmte Kleidungsstücke oder Stilrichtungen mag, sich aber wegen gesellschaftlicher Zwänge nicht traut, so etwas zu tragen, dies im Spiel ausleben. Ein Beispiel dafür ist für eine Frau ein hautenger Lederanzug, in dem viel Haut sichtbar ist. Dies würde, im Alltag getragen, mit Sicherheit zu einigen verachtenden Blicken und Bildung einer negativen Meinung über diese Person führen. Auch gibt es oft Dinge, die die Menschen an ihrem Erscheinungsbild nicht mögen. Dies kann zum Einen, die durch den Beruf aufgezwungene Kleidung, aber auch der Körper direkt sein. So gibt es wohl viele Menschen, die sich wünschen, etwas weniger zu wiegen, oder kleinere Menschen, die gerne größer wären als sie es sind. In Spielen haben diese Menschen die Möglichkeit, ihr Erscheinungsbild selbst zu bestimmen. Hierzu haben die Umfragen von Nicholas Yee (vgl. [Yee03b]) gezeigt, dass etwa 35% der männlichen und 25% der weiblichen Spieler ihren Avatar als eine idealisierte Version von sich selbst sehen. Im Umkehrschluss stellt sich die Frage, wie viele der Befragen ihren Avatar als eine reine Spielfigur ohne einen Bezug zu ihrer Persönlichkeit sehen. Dies ist bei ca. 40% der männlichen und 25% der weiblichen Spieler der Fall.

### A.2.5 Psychologie

Avatare haben oft eine subtile Bedeutung. John Suler hat eine Studie zur den psychologischen Aspekten eines Avatars in "The Palace" durchgeführt (vgl. [Sul01]). "The Palace" ist eine virtuelle 2D Umgebung. In ihr sind der Gestaltung der Avatare fast keine Grenzen gesetzt. Es gab bestimmte Richtlinien, die unter Anderem besagten, dass sie jugendfrei seien sollen.

Von Anfang an ist schon ein Satz von Standardavataren dabei. Diese bestehen aus simplen Smileys, bei denen die Benutzer die Gesichtsfarben ändern und Accessoires, wie einen Hut oder auch eine Bierflasche, hinzufügen können. Es sind einfache Avatare, doch

man kann mit ihnen Gefühle und Verhalten ausdrücken. Aufgrund der Tatsache, dass nur Nutzer, die ihre Software registriert haben, sich eigenen Avatare erstellen können, und unerfahrene es des öfteren nicht geschafft haben, sich eigene zu erstellen und diese anzuzeigen, werden diese Standardavatare mit neuen, unerfahren Nutzern, so genannten Newbies, assoziiert, die nicht mit der Kultur von "The Palace" vertraut sind und sich erst noch etablieren müssen.

Die Avatare, die von den Benutzern selbst erstellt werden, wurden von Suler anhand dessen, was sie visuell darstellen, in verschiedene Kategorien eingeteilt. Diese können wiederum psychologische Bedeutungen haben.

**Tiere**
Tiere stehen für verschiedene Eigenschaften. Ein Bär ist stark, ein Hund treu und ein Adler frei, um nur einige Beispiele zu nennen. Als Avatar einer Person können sie für Eigenschaften stehen, die die Person mit sich selbst verbindet oder gerne verbinden würde.

**Zeichentrickfiguren**
Eine Comicatmosphäre führt dazu, dass Menschen zu Dingen, wie Probleme auf der Arbeit, Abstand nehmen können. Die Menschen neigen dazu, Zeichentrickfiguren zu wählen, mit denen sie sich identifizieren können oder die Eigenschaften verkörpern, die ihnen fehlen und die sie gerne hätten. Zum Beispiel steht der Genie von Aladin für einen mächtigen und hilfsbereiten Freund und Bugs Bunny für einen freundlichen Hochstapler.

**Berühmte Persönlichkeiten**
Dieser Typ von Avatar ist ein Trendspiegel. Die Avatare einer berühmten Person sprießen aus dem Boden wie Unkraut und, wenn der Trend wieder vorbei ist, verschwinden sie recht schnell von der Bildfläche. Die Menschen benutzen solche Avatare, um persönliche Eigenschaften, die man mit der entsprechenden Berühmtheit verbindet, auszudrücken. Diese Eigenschaften können auch wieder die sein, die man selber hat, oder die, die man gerne hätte. Teilweise machen sie sich aber auch über eben diese Eigenschaften der Person lustig. Manche versuchen durch das zeigen ihrer Verbundenheit mit dem Star ihre Selbstachtung zu stärken. Andere Avatare haben den Sinn, die Vorlieben und Interessen des Benutzers zu zeigen. So präsentiert er seine Persönlichkeit und kann leichter Gleichgesinnte finden.

**Böse Avatare**
Diese Art von Avatar ermöglicht es den Menschen, ähnlich wie Halloweenkostüme, ihre schwarze Seele auszuleben und sie zu zeigen. Andere versuchen so ihre aktuelle Stimmung, Niedergeschlagenheit oder Wut auszudrücken. Weiter kann solch ein Avatar auch der Kommunikation dienen. Ein kurzer Wechsel zu einem Totenschädel kann einer anderen Person unmissverständlich zu verstehen geben, dass er die Person in Ruhe lassen soll.

**Reale Gesichter**
Nur wenige Palace-Besucher verwenden ein Bild von sich selbst als Avatar. Doch auch eigenen Photos werden benutzt, um Freunden zu zeigen, wie man in der Realität aussieht, doch oft nicht darüber hinaus. Dies liegt vor allem daran, dass die meisten anonym bleiben wollen oder neue Identitäten ausprobieren möchten. Das zeigen seines realen Gesichtes ist ein Zeichen von Zutrauen, Ehrlichkeit, Einheit, Freundschaft und auch von Romantik.

**Erkennungszeichen**
Es existieren Avatare, die direkt mit einem speziellen Benutzer verbunden werden. Diese Avatare sind einzigartig und oft besonders kreativ, aber gelegentlich auch sehr schlicht. Normalerweise sind diese Avatare unverwechselbar, oder sollen es zumindest sein. So müssen Benutzer, die versuchen, solche Avatare zu kopieren, mit starker Kritik rechnen. Diese Avatare sind "Markenzeichen" der entsprechenden Person und sind der Inbegriff der virtuelle Identität.

**Passend zu einer Umgebung**
Manche Avatare werden speziell für besondere Umgebungen, wie Wasser oder Luft, Räume oder sogar für einzelne Positionen in Räumen, wie einen speziellen Stuhl, erstellt. Sie zeugen von ausgesprochener Kreativität des Nutzers und sind ein Statussymbol. Sie verdeutlichen, dass sich die Person gut in der Palace-Umgebung und mit ihrem Computer sowie dem Programm auskennt, was auf einen erfahrenen Nutzer schließen lässt.

**Avatare mit Superkräften**
Viele Menschen hätten gerne besondere Fähigkeiten. Solche Fähigkeiten werden oft durch Superhelden, wie sie in Comics vorkommen, personifiziert. Eben diese werden als Avatare genutzt, hauptsächlich von männlichen Jugendlichen. Das ständige verwenden eines Super-Avatars ist ein Zeichen für Unsicherheit und Hilflosigkeit, denen man mit solchen Superkräften entkommen könnte. Mit diesen Avataren werden häufig regelrechte Machtkämpfe ausgetragen, die darauf beruhen, wer den mächtigeren Avatar kreieren kann.

**Erotische Avatare**
Erotische Avatare treten häufiger in weiblicher als in männlicher Form auf. Des öfteren sind es sogar Männer, die einen weiblichen Avatar benutzten. Manche versuchen zu flirten, bei wenigen ist es Bereitschaft zu Cybersex, doch die meisten Personen versprechen sich durch ihre Avatare lediglich Aufmerksamkeit, die sie in der Regel auch bekommen. Aufreizende Avatare haben eine besonders anziehende Wirkung auf andere Palace-Benutzer. Bei vielen Menschen trifft es in diesem Fall zu, dass es die oberflächlichen Dinge sind, die sie anziehen. Sie mögen einfach die Vorstellung, sich mit einer attraktiven Person zu unterhalten.
Wie bei den Superhelden gibt es auch unter den erotischen Avataren einen stillen Wettbewerb, wer den attraktivsten Avatar kreieren kann. Einige Nutzer trachten nach dieser Bewunderung, die man ihnen entgegen bringt, wenn sie solche Avatare kreieren können.

**Ungewöhnliche/schockierende Avatare**
Mit diesen Avataren wollen die Menschen andere überraschen, sich über etwas lustig machen oder provozieren. Sie werden meistens von jugendlichen verwendet, die so ihre Selbständigkeit und ihre Individualität ausdrücken und ihre Grenzen ausloten möchten.

**Abstrakte Avatare**
Abstrakte Avatare, wie Farbwirbel, sind ein Zeichen, dass die Person symmetrische Formen mag, ein konzeptioneller Denker ist oder ihre künstlerische Ader auslebt.

**Reklame**
Einige Benutzter verwenden ihre Avatare dazu, um Botschaften zu verbreiten. Bei diesen kann es sich um politische oder philosophische Ansichten handeln. Die Liste dessen, was man auf diese Weise alles kundtun kann, ist beliebig lang. Auch die Anzahl der Möglichkeiten, wie es im Avatar geäußert wird, ist unendlich. Ein einfaches Beispiel ist eine Kuh, die ein Schild mit der Aufschrift "Ich bin Vegetarier" in die Höhe hält.

**Zugehörigkeit**
In Motorradclubs tragen in der Regel alle Mitglieder eine Jacke oder eine Weste, auf deren Rückseite das Zeichen ihres Clubs zu sehen ist. Dies ist ein Erkennungsmerkmal und drückt die Zugehörigkeit zu einem Club aus. Genauso verhält es sich mit Gruppen

in “The Palace”. Wenn alle Mitglieder den selben oder einen sehr ähnlichen Avatar verwenden, wird so die Zugehörigkeit zu einander signalisiert. Auch unterschiedlich aussehende Avatare, die aber alle einen bestimmten Aspekt gemeinsam haben, drücken dies aus. Ein Beispiel sind verschiedene Tiere, die aber immer eine schwarze Lederjacke und Sonnenbrille tragen.

## A.3 Kommunikation

### A.3.1 Welche Kommunikationsmöglichkeiten gibt es

Wenn man an Kommunikationsmöglichkeiten in Computerspielen denkt, fällt einem als Erstes ein Textchat[3] ein, der in den meisten Spielen integriert ist. Etwas bequemer ist das Nutzen von “Voice over IP” (siehe Kapitel A.3.2). Doch neben dieser wortbasierten Kommunikation beschreiben Tony Manninen und Tomi Kujanpää (vgl. [MK05]) die Möglichkeiten über Avatare, besonders mit deren Gesichtern, durch nicht wortbasierte Töne und Kinesik[4] zu kommunizieren.
Neben den schon in Kapitel A.2.2 beschriebenen Informationen, die durch den Avatar kommuniziert werden, birgt besonders der Gesichtsausdruck eine große Ausdruckskraft. In “Battlefield 1942” ist dieser leider nicht direkt durch den Spieler steuerbar sonder an den Gesundheitszustand und andere Spielhandlungen, wie wenn der Avatar verletzt wird oder vordefinierte Befehle gibt, gekoppelt. Die Kinesik bietet nicht nur Aufschluss über die aktuelle Handlung des Spielers (feuern, zielen) sondern kann auch kontextabhängig gedeutet werden. Wenn eine Spielrunde gewonnen ist, ist das aufspringen eines Avatars aus seiner Deckung als Jubel zu verstehen. Weiter kann durch bestimmte Bewegungen kommuniziert werden. In “Battlefield 1942” hat der Spieler auch keine direkte Kontrolle über die Körperteile, so dass es nur vorgenerierte Gesten, ein Daumen-hoch Zeichen und zwei militärische Handzeichen, möglich sind.
In dem Spiel “Counter Strike” gibt es die Möglichkeit, persönliche Spraylogos an Wände zu sprühen. Diese Logos können ein persönliches Erkennungszeichen sein oder anderen Spielern Informationen, wie Interessen, Vorlieben und politische Ansichten, mitteilen. Darüber hinaus haben es die Spieler geschafft, die im Spiel eingebauten Mechanismen, die eigentlich nicht zur Kommunikation gedacht sind, für eben diese zu missbrauchen. Tote Spieler können eigentlich nicht mit den lebenden reden, damit diese keine Informationen über Standorte der Gegner preisgeben können. Doch in “Counter Strike” gibt es die Möglichkeit, über die als nächstes zu bespielende Karte[5] abzustimmen. Indem man als Toter anstelle eines Kartennamens einfach eine Nachricht für andere Spieler eintippt, kann diese Nachricht von allen Spielern gelesen werden (vgl. [WBB02]).
John Suler beschreibt am Beispiel von “The Palace” (vgl. [Sul01]) ähnliche Kommunikationsmöglichkeiten. So kann durch das wilde umherhüpfen eines Avatars im Raum Freude und durch das nahe bei einander stehen von zwei Avataren Freundschaft (oder mehr) ausgedrückt werden. Weiter gibt es die Option, durch das wechseln zu einem bestimmten Avatar, bestimmte Tätigkeiten oder Gefühle auszudrücken. Ein Totenschädel symbolisiert dem Gesprächspartner, dass er verschwinden soll. Ein Hund mit einem Beutel an einem Stock, den er über die Schulter legt, kann ein Zeichen dafür sein, dass man “The Palace” verlässt und offline geht. Zum Schluss sei noch erwähnt, dass ein eigens erstellter Avatar in einer virtuellen Umgebung immer ein gutes Gesprächsthema ist, besonders für Menschen, die sich das erste mal begegnen. Es ist, wie in der Realität über das Wetter zu reden, nur dass die Menschen stärker mit ihren Avataren verbunden sind als mit dem Wetter.

[3] Mitteilungen als Text per Tastatur eingeben und verschicken.
[4] Kinesik ist ein Teil der Kommunikationswissenschaft, der sich mit der kommunikativen Relevanz von Bewegungen beschäftigt.
[5] Karten sind Spielumgebungen. In “Counter Strike” kann man unter Anderem in Hochhäusern, Kaufhäusern oder Stadtvierteln spielen.

In einigen Strategiespielen, darunter "Freeciv", ein OpenSource "Civilization" Clone, und "Diplomacy", haben die Spieler die Möglichkeit, mit anderen Spielern, darunter auch die NPCs[6], in Verhandlungen zu treten. In "Freeciv" ist dieses sehr einfach gehalten. Der Antragsteller erstellt eine Liste mit Angeboten und dafür verlangten Gegenleistungen. Eine solche Liste kann unter anderem Waffenstillstände, Alianzen, den Austausch von Forschungergebnissen, Geld und das zur Verfügungstellen der Landkarten beinhalten. Ein Spieler, dem eine solche Liste unterbreitet wird, kann diese annehmen, ablehnen oder verändern. Verändern bedeutet, dass Punkte von der Liste gestrichen und neue hinzugefügt werden können. In jedem Fall sind die Punkte, die einer Liste hinzugefügt werden können, aus einem Menü auszuwählen. Es kann kein Freitext verwendet werden. "Diplomacy" geht noch einen Schritt weiter. Hier kann man beliebige Spielzüge mit den anderen Spielern, inklusive der NPCs, absprechen. Dabei verschiebt man alle an dem Spielzug beteiligten Spielfiguren (auch die des anderen Spielers) simuliert und schlägt den so entstehende Spielzug dann dem anderen Spieler vor. Auf diese Weise ist es ebenfalls möglich, einen anderen Spieler um Erlaubnis zu Fragen, einen Spielzug auszuführen, der gegen ein bestehendes Abkommen verstoßen würde, wie etwa das Durchqueren einer entmilitarisierten Zone. Weiter hat man die Option, Angebote mit einem Ultimatum zu versehen, um den Verhandlungspartner dazu zu zwingen, das Angebot zu akzeptieren.
Zuletzt sei hier noch der Adressatenkreis von Nachrichten erwähnt. Bei teambasierten Shootern[7] kann man Nachrichten entweder an alle Spieler senden oder nur an Spieler aus seinem Team. Bei dem MMORPG[1] "World of Warcraft" können Spieler flüstern, damit nur ein einzelner Spieler die Nachricht erhält, etwas sagen, dann können nur Spieler, deren Avatar sich in einer bestimmten Umgebung um den sprechenden Avatar herum befinden, die Nachricht lesen, oder sich über spezielle Chatkanäle verständigen, über die nur eine ausgewählte Gruppe von Spieler mit einander kommunizieren kann (vgl. [Tho07], S. 175).

### A.3.2 Voice over IP

VoIP bietet Spielern die Möglichkeit, mit Hilfe von spielexternen Programmen und einem Mikrophon, mit ihrer eigenen Stimme zu kommunizieren. Ein langwieriges Tippen von Nachrichten wird überflüssig. Nicholas Yee (vgl. [Yee06]) hat in seiner Umfrage bei über 3.000 Spielern von MMOGs[8] herausgefunden, dass ca. 70% der Befragten schon einmal VoIP genutzt haben. Bei einer weiteren Umfrage unter mehr als 2.400 Spielern ergab sich, dass ca. 40% der Befragten VoIP häufig oder immer nutzen, und dass ca. 60% es als sehr hilfreich empfinden.
VoIP bietet eine schnellere Kommunikation als der Textchat, wodurch eine stärkere Koordination bei Angriffen im Spiel möglich ist. Deshalb ist VoIP auch stärker in Gruppe vertreten, die ihre Charaktere schnell mächtiger machen oder die schwierigere Kämpfe meistern wollen. Ein zweiter Aspekt, der durch VoIP gefördert wird, ist der soziale. Das sprechen mit Freunden ist unkomplizierter und VoIP kann zu einem stärkeren Zusammenhalt führen. Leider kann durch die Nutzung von VoIP auch genau das Gegenteil passieren. Es kann eine vormals enge Gruppe in zwei Teilgruppen aufsplitten. Die, die VoIP-Programme haben, und die, die sie nicht haben. Der dritte und ein negativer Aspekt von VoIP ist, dass Illusionen zerstört werden. Die Stimme eines Ogers, die man sich im Kopf ausdenkt, klingt immer besser als die Stimme, die ein Gegenüber versucht zu imitieren (falls er sich überhaupt die Mühe macht). Mit der Stimme werden

[6] Non Player Character (Nichtspieler Charakter): Ein Charakter, der nicht von einem Spieler sondern durch den Computer gesteuert wird.
[7] Shooter sind Spiele, in denen es darum geht, seine Gegner mit Waffen zu töten. Sind sie teambasiert, handelst es sich um mehrere Gruppen, die sich gegenseitig bekämpfen. Dies steht im Gegensatz zu Spielen oder Spielmodi, in denen Jeder gegen Jeden kämpft.
[8] Massive(ly) Multiplayer Online Games, ein Computerspiel für eine große Anzahl an Mitspielern, welches über das Internet gespielt wird.

zusätzlich soziale Merkmale wie Geschlecht, Alter und Herkunft preisgegeben und es gibt Spieler, die so etwas gerne geheim halten würden. Ein 16 Jahre alter Spieler, der eine Gruppe anführt, wird seinen Mitspieler nicht gerne sein Alter verraten.

### A.3.3 Über was reden Spieler und wozu

#### Diskussionsthemen

Talmadge Wright, Eric Boria und Paul Breidenbach (vgl. [WBB02]) haben eine Studie über die Kommunikation in dem FPS[9] "Counter Strike" durchgeführt. Sie unterteilen das, worüber in dem Spiel geredet wird, in fünf Kategorien, von denen "Konflikte im Spiel" und "Spielleistungen und -geschehen" am häufigsten im Spiel auftreten:

**Kreativität im Spiel**
Diese Kategorie wurde von den Autoren besonders untersucht und in weitere fünf Subkategorien aufgeteilt:

> **Namen und Identität**
> Jeder Spieler kann sich einen eigenen Namen geben. Die Möglichkeiten der Namensgebung ist unendlich und soll hier nicht näher betrachtet werden. Diese individuellen Namen bieten ein großes Potential zu Wortspielen. Die Autoren beschreiben einen Fall, bei dem in einer Gruppe von Spielern alle Tiernamen gewählt haben. Darunter waren Wombats, Hühner, Enten und Kühe. Es entstand eine Diskussion, welche Art von Bauernhof Wombats hält. Diese Diskussion führte dazu, dass die Spieler die Laute, die die Tiere, nach denen sie sich benannt hatten, machen, schrieben und sich auf diese Weise "unterhielten".
>
> **Witze, Ironie und Wortspiele**
> Wortspiele werden sehr oft in Diskussionen verwendet. Neben dem Erreichen des eigentlichen Spielzieles ist das eine weitere Möglichkeit, um sich selbst zu profilieren. Die Autoren beschreiben zwei Fälle hiervon. In dem einen Fall wird die Zweideutigkeit des Wortes "rounds" (zu Deutsch Spielrunde oder Projektil) verwendet. Ein Spieler muntert einen anderen Spieler, der schlecht spielt, auf, indem er ihm sagt "you'll be ok once ya get a couple rounds in ya" (Nach ein paar Runden wirst du besser), worauf er mit "I have had a couple rounds in me, mostly from a certain person's AK 47." (Ich habe ein paar Kugel in mir, die meisten sind von jemandes AK 47[10]). Beim zweiten Fall wird die Eigenschaft einer Karte[5] für den Witz ausgenutzt. Es handelt sich um ein im Bau befindliches Hochhaus, welches durch die Terroristen gesprengt werden soll. Spieler können von diesem Gebäude fallen oder absichtlich hinunter springen. Diese Eigenschaft wird von einem Spieler, der im Spiel keine guten Leistungen erbracht hat, dazu verwendet, einen virtuellen Selbstmord zu begehen, bevor er das Spiel verlässt. Weiter werden sehr oft Beschimpfungen und Sticheleien durch das Hinzufügen von Symbolen, die Ironie ausdrücken, entschärft. Ein zwinkerndes Smiley ;-) ist ein solches Symbol. Die Beschimpfungen beziehen sich dann auf die Spielsituation, zum Beispiel einen vorausgegangen Tod, und dienen nicht dem Denunzieren des anderen Spielers.
>
> **Karten und Logos**
> Neu Karten[5] können von jedem erstellt werden. Die Qualität und die Eigenschaft von neuen Karten werden von Spielern oft diskutiert. Sie werden

[9] First Person Shooter - Ein Spiel, in dem man die Welt mit den Augen des Avatars sieht (first person) und in dem man Gegner mit Handfeuerwaffen bekämpft (shooter).
[10] Die AK 47 ist ein russisches Sturmgewähr.

danach bewertet, ob sie für beide Seiten fair sind, keine Seite einen Vorteil hat und ob Taktiken und Strategien erarbeitet werden müssen, damit eine Seite auf dieser Karte gewinnen kann.
Wie in Kapitel A.3.1 erwähnt wurde, können sich Spieler in "Counter Strike" Spraylogos erstellen und sie an Wände sprühen. Diese Logos dienen selber als Kommunikationsmedium und über sie wird auch geredet. Es wird nicht nur auf ihren Inhalt Bezug genommen, sonder auch auf Qualität und die Komplexität des Bildes. Über diese Aspekte werden Kritik und Lob geäußert sowie Tipps und Verbesserungsvorschläge zu ihnen gegeben.

**Spielregeln verändern**
In "Counter Strike" können einige Spielparameter verändert werden. Darunter fallen unter Anderem die Rundenzeit oder die Zeit, nach deren Ablauf zur nächsten Karte gewechselt wird, aber auch die Erdanziehungskraft. Über diese Parameter wird selbstverständlich gesprochen und sie werden, bei Bedarf, durch den Spieladministrator angepasst.

**Anspielungen auf populäre/kulturelle Sachverhalte**
Die letzte Kategorie der Kreativität im Spiel ist das Verbinden von Spielinhalten mit populären Fernsehserien oder anderen Sachverhalten des öffentlichen Interesses. Oft ist der Ausruf "doh!", welcher von Homer Simpson in der Fernsehserie "Die Simpsons" benutzt wird (in der deutschen Version ist es "Nein"), wenn er wieder etwas vermasselt hat, zu vernehmen. Ein weiteres Beispiel ist eine Spielszene, in der ein Spieler mit den Namen "Kenny" getötet wird. In der Fernsehserie "South Park" gibt es einen Akteur mit diesem Namen, der im Laufe jeder Folge stirbt. Wenn dies passiert, wird dieses Geschehen immer durch den Satz "Oh, my god, they killed Kenny." (Oh mein Gott, sie haben Kenny getötet) von einer anderen Figur kommentiert. Eben dieser Satz wurde, nach dem Tod Kennys, im Spiel auch verwendet.

**Konflikte im Spiel**
Spieler, die sich das ganze Spiel an einer Stelle aufhalten, sich verstecken und darauf warten, dass ein Gegenspieler vorbeikommt, den sie dann erschießen können, sind bei vielen Spielern nicht sehr beliebt. Dieses Missfallen wird, wenn ein so genannter "Camper" mitspielt, im Chat zum Ausdruck gebracht und es wird über diesen Sachverhalt diskutiert. Noch verrufener sind Schummler. Diese benutzen zum Beispiel Headshootskripts[11]. Wenn vermutet wird, dass ein Spieler schummelt, oder so etwas entdeckt wird, wird dies auch den anderen Spielern verkündet und darüber diskutiert, was meist in Form von Anschuldigungen und Rechtfertigungen geschieht. Weitere Gesprächsthemen in dieser Kategorie sind die AWP, ein Scharfschützengewähr mit enormer Durchschlagskraft, mit dem man einen Spieler durch einen Schuss in den Torso töten kann, und das Verbannen von Spielern aus dem Spiel durch Spieladministratoren.

**Beschimpfungen**
Unter Spielern sind Beschimpfungen nichts ungewöhnliches. Teilweise begründen sich diese in einer Unzufriedenheit mit der Spielweise oder den Fähigkeiten eines Mitspielers oder in der Unterlegenheiten eines Mitspielers gegenüber einem Gegenspieler. Die Beschimpfungen fangen bei kleinen Sticheleien an und gehen bis zu geschlechtsdiskriminierenden und rassistischen Äußerungen. In der Regel haben sie jedoch einen spielhaften Charakter und sind weder ernst gemeint noch werden sie als beleidigend aufgefasst (vgl. [Tho07], S. 176).

**Spielleistungen und -geschehen**
Ein wichtiger Punkt ist natürlich das Spiel an sich. Es wird viel über Spieltaktiken diskutiert. Was falsch gemacht wurde, was man vielleicht besser machen könnte und wie

[11] Ein Programm, welches dafür sorgt, dass man immer den Kopf des Gegner trifft, auf den man zielt. Durch einen Kopfschuss wird ein Spieler, egal mit welcher Waffe er ausgeführt wird, sofort getötet.

man es als nächstes versucht. Hierzu zählt auch die Abstimmung der Spieler im Geschehen, um Taktiken anzupassen oder zu erfahren, wo sich ein Mitspieler gerade befindet. Des Weiteren wird sich, für Fehler die gemacht wurden, bei den Mitspielern entschuldigt und es werden Spieler, die besonders gut spielen, für ihre Handlungen gelobt. Dazu werden auch Spielstatistiken diskutiert und ausgewertet. Unerfahrene Spieler stellen Fragen und erhalten Hilfestellung. Tote Spieler unterhalten sich über die Handlung der noch lebenden Spieler. Darüber hinaus gibt es Diskussionen zwischen den Spielparteien, in denen sich für das gute Spiel bedankt, über eine Revanche gesprochen oder die Ausgewogenheit der beiden Parteien und deren Fairness thematisiert wird. Diesbezüglich wird über den Wechsel von Spielern gesprochen, damit beide Teams gleich stark sind, oder entschieden, in welches Team ein neuer Spieler am besten gehen sollte. Am Ende einer Spielrunde kann auch beobachtet werden, dass zwischen den letzten verbleibenden Spielern die Kommunikation zu einer psychologischen Waffe wird, in dem man versucht, den anderen zu verunsichern.

**Technik und spielunabhängige Diskussionen**
Selbstverständlich wird auch über die technische Seite des Spiels geredet. Wie ist die Verbindung zum Server[12], wie ist die Graphik des Spiels, wie gut ist welche Karte[5] und welche soll als nächstes gespielt werden. Darüber hinaus wird über Computer geredet. Welche Computer werde zum Spielen genutzt, welche Hardware besitzt man und welche benötigt man für welches Spiel. Genauso wird über andere Spiele geredet sowie darüber, wer einem Clan[13] beitreten darf, wem man einen Beitritt anbieten sollte und wer der Beste im Clan ist.

Neben diesen Diskussionsthemen hat Nicholas Yee (vgl. [Yee02]) herausgefunden, dass auch über persönliche Angelegenheiten geredet wird. Vorallem Jugendliche neigen dazu, mit ihren Onlinefreunden über Dinge zu reden, über die sie mit niemanden in der Realität sprechen. In Zahlen ausgedrückt, hat seine Umfrage mit über 3.400 Spielern von 5 MMORPGs[1] ergeben, dass ca. 23% der Befragten schon persönliche Angelegenheiten und Geheimnisse mit ihren Onlinefreunden geteilt haben, die sie nicht ihren Freunden in der Realität anvertrauen. Dabei tendieren Frauen stärker zu diesem Verhalten als Männer. Dies liegt daran, dass es für viele Menschen leichter ist, mit Leuten zu reden, die sie noch nie gesehen haben. Auf diese Weise spenden Spiele, weil es immer jemanden gibt, mit dem man reden kann, Trost in einer sicheren Umgebung. Das erklärt auch, dass ca. 40% von über 3.300 Probanden einer anderen Umfrage angaben, dass einige ihrer Onlinefreunde ebenso gut oder sogar besser sind als ihre Freunde in der Realität.

### Die Funktion der Kommunikation

Jan-Noël Thon unterteilt die Funktion von Kommunikation in dem Shooter "Halo" und dem MMORPG[1] "World of Warcraft" in ludische, soziale und narrative (vgl. [Tho07], S. 175ff). Kommunikation mit ludischer Funktion ist hauptsächlich taktischer und strategischer Natur. Charakteristisch ist, dass das aktuelle Spielgeschehen thematisiert wird und nicht vorangegangene Spiele analysiert werden. Soziale Funktion haben Gespräche, die quasi außerhalb des Spiels geführt werden, auch wenn sie über das Spiel handeln und über Kommunikationswege aus dem Spiel geführt werden. Beispiele sind Prahlereien nach gelungenen Spielaktionen, Lob für Spielleistungen oder Gespräche über andere Freizeitaktivitäten. Kommunikation mit narrativer Funktion findet man nicht in Shootern, dafür aber in "World of Warcraft" und allgemein in Rollenspielen. Im Gegensatz zur Kommunikation mit sozialer Funktion, die sich auf Spieler oder deren Online-Identität

[12] Ein entfernter Computer, auf dem das Spiel läuft. Die Spieler melden sich von ihrem Rechner aus auf diesem Server an, um auf ihm zu spielen.

[13] Ein Clan ist eine Gruppe von Spielern, die im Verband zusammen Computer spielen, regelrecht trainieren, gegen andere Clans bei Wettkämpfen in Computerspielen antreten und teilweise selber Wettkämpfe veranstalten.

bezieht, bezieht sich Kommunikation mit narrativer Funktion auf die Avatare, die Figuren in einer komplexen fiktionalen Welt sind. Solche Gespräche sind Rollenspiele. Sie handeln von Geschehnissen im Spiel, die als Geschichte erzählt werden, oder sind erfundene Geschichten, die in die Spielwelt passen und in dieser passiert seien könnten.

Zum Schluss dieses Kapitels sei noch erwähnt, dass die Spieler eine eigene Sprache benutzen, die man überall im Internet, unter anderem auch in Chats und Foren, findet. Es werden so genannte Emoticons, das sind mit ASCII-Zeichen erstellt Smileys ( " ;-) ", " :-) ", " 8-) " ), oder Emotes, was in Sternchen gesetzte, mit Wörtern ausgedrückte Emotionen sind (" *traurig* "), eingesetzt. Teilweise werden hierbei auch Abkürzungen benutzt (" *rofl* " steht für "rolling on the floor, laughing" - sich vor Lachen auf dem Boden rollen). Abkürzungen spielen bei Textchats unabhängig von emotionalen Ausdrücken eine große Rolle, da das Schreiben länger dauert als das Sprechen oder das Lesen. Ein Beispiel hierfür ist "afk", was für "away from keyboard" (zur Zeit nicht an der Tastatur) steht.

### A.3.4 Chatbots

#### Wie arbeiten sie

In diesem Kapitel möchte ich zunächst kurz am Beispiel des Open-Source-Chatbot A.L.I.C.E.[14] beschreiben, wie Chatbots arbeiten. Für genauere Informationen sei auf Kapitel 13 des Buchs "Web-Kommunikation mit OpenSource" von Claus Möbus et al. (vgl. [FJS06]) verwiesen.

Eingaben an den Bot werden als erstes in Sätze und Satzglieder unterteilt. Dies wird durch Orientierung an der Punktierung erreicht. Anschließend werden die Wörter normalisiert. Hierbei werden Synonyme zu Wörtern bestimmt ("bereits" wird zu "schon") und die Wörter in ihre Grundform überführt (aus "Häuser" wird "Haus"). Für Ersteres ist die Verwendung eines Wörterbuches unerlässlich, während die Grundformbildung rein algorithmisch durch Abschneiden von Endung und Umwandeln von Umlauten (ä wird zu a) erfolgen kann. Allerdings ist auch hierbei das Benutzen eines Wörterbuches vorteilhaft, damit die Umwandlung von "schön" zu "schon" verhindert werden kann.
Die normalisierte Anfrage wird danach mit dem Wissen in der Wissensdatenbank verglichen. Die Wissensrepräsentation erfolgt in AIML, einer XML-Spezifikation. Die Anfrage wird mit Anfragemustern, die auch Wildcards[15] enthalten können, verglichen. Dies geschieht zuerst mit möglichst spezifischen Mustern, was bedeutet, dass sie keine Wildcards und gegebenenfalls eine Referenz auf das Gesprächsthema enthalten. Das Wissen wird zur Laufzeit in einem Baum abgespeichert, so dass eine Eingabe Wort für Wort abgearbeitet werden kann. Hierbei wird im Baum immer zuerst im spezifischten Pfad gesucht. Führt das zu keiner Übereinstimmung mit der Eingabe, wird mittels Backtracking[16] dann der nächst unspezifischere Pfad abgearbeitet. Wurde dann ein zur Eingabe passendes Muster gefunden, ist diesem Muster im einfachsten Fall nur eine Antwort zugeordnet, die ausgegeben wird. Wenn dem Muster mehrere Antworten zugeordnet sind, wird zufällig eine ausgewählt. Zusätzlich kann es sein, dass das Gesprächsthema, der Satz oder Satzteile, wie ein Eigenname, abgespeichert werden, um später oder direkt in der aktuellen Antwort darauf Bezug zu nehmen. Weiter ist es möglich, dass an dieser Stelle Skriptsprachen zum Einsatz kommen, damit der Chatbot Rechnungen ausführen kann.

[14] The A.L.I.C.E. Artificial Intelligence Foundation, siehe http://alicebot.org/, letzter Zugriff am 16.03.08

[15] Ein spezielles Zeichen. Es dient als Platzhalter und an seiner Stelle kann ein beliebiges Wort stehen.

[16] Es wird auf dem bisherigen Suchpfad ein oder mehrere Knoten zurück gegangen, um einen anderen Pfad zu untersuchen.

#### Warum wirken sie menschlich und wie entlarvt man sie

Ein Chatbot übernimmt oft eine bestimmte Rolle. Indem er immer auf die selbe Art und Weise, die seiner Rolle entspricht, mit Eingaben umgeht, versucht er, menschlich zu wirken. Durch einen schematischen Charakter halten Menschen einen Bot zumindest kurze Zeit für einen Menschen, wenn er denn die Rolle, und somit die Eigenschaften, übernimmt, die sich eine Person vorstellt und seinem Gegenüber zuschreibt.
ELIZA, der wohl berühmteste Chatbot, von Joseph Weizenbaum übernimmt die Rolle eines Psychiaters. ELIZA beantwortet keine Fragen, sie stellt immer welche. Auch wenn die Eingabe eine Frage ist, stellt ELIZA eine Gegenfrage, genau wie ein Psychiater.
Das Programm "Parry" von Kenneth Colby verwendet einen anderen Ansatz. Parry übernimmt die Rolle eines paranoiden Gesprächspartners und reagiert auf Eingaben paranoid und aggressiv.
"Elbot" von Fred Roberts hingegen ist immer sarkastisch und versucht, seinen Gegenüber so zu Unterhalten.
Der oben beschriebene Bot A.L.I.C.E. rettet sich ebenfalls durch Witz aus einer schwierigen Lage. Sie Antwortet auf die Frage "Was ist der Unterschied zwischen natürlicher Sprachverarbeitung und professioneller Kosmetikwissenschaft?" mit "Sind sie denn nicht gleich?"

Die Grenzen von Chatbots werden leider noch recht schnell erreicht. Einige stellen mehrmals die gleiche Frage, was ein Mensch nur selten tut. Weiter können sie Eigennamen nur anhand des Satzkontextes erkennen, was bedeutet, dass sie die Eigennamen nicht immer erkennen und zum anderen auch unsinnige Eingaben als Eigenname annehmen können. Woher sollte ein Bot auch wissen, dass "EQERQER" kein Namen ist, wenn ein Benutzer "Ich heiße EQERQER" schreibt? Oder das "Ham" in "West Ham College" nicht "Schinken" bedeutet? Ähnlich verhält es sich mit Rechtschreibfehlern. Falsch geschriebene Wörter können von einem Bot nur schwer korrigiert werden, um sie richtig weiter zu verarbeitet. Auch ein Gesprächskontext wird von Chatbots nicht immer richtig erkannt, wenn ein Bot überhaupt versucht, diesen zu ermitteln und abzuspeichern. Die wenigsten Bots wissen, wer mit "er" gemeint ist, wenn dieser ein paar Eingaben zuvor namentlich genannt wurde. Auch Ironie und Wortspiele können von einem Bot kaum als solche entlarvt werden. Ebenfalls bereiten Fragen nach Unterschieden Probleme. Bei obigem Beispiel "Was ist der Unterschied zwischen natürlicher Sprachverarbeitung und professioneller Kosmetikwissenschaft?" müssen zuerst einmal drei Stichworte, Unterschied, Sprachverarbeitung und Kosmetikwissenschaft, erkannt werden. Viele Bot sprechen aber nur auf ein Schlüsselwort, die Sprachverarbeitung, an. Anschließend muss in der Wissensdatenbank zu diesem Unterschied auch eine Antwort abgelegt sein. Ein weniger schwerwiegender Fehler ist Fachvokabular. Ein allgemeiner Chatbot kann nichts mit fachspezifischen Spezialausdrücken anfangen. Da diese in der Regel auch nur in einem bestimmten Kontext verwendet werden, ist es wahrscheinlich, dass nur ein spezieller Chatbot diese Wörter kennen muss.

### A.3.5 Automatische Textgenerierung

Marcus Lechner schildert in seinem Vortrag "Automatische Textgenerierung in Computerspielen" (vgl. [Lec06]) denn Sinn und die Möglichkeiten einer automatischen Textgenerierung, die sich an den Handlungen des Spielers orientiert. Dialoge von NPCs[6] oder Zusammenfassungen können so unterhaltsamer, abwechslungsreich und menschlicher werden. Statistiken, die teilweise mit einem unübersichtlichen Zahlenwust einhergehen, würden durch einen unterhaltsame Geschichte ersetzt.
Teilweise finden sich solche Textgeneratoren schon heute in Spielen. In "Genforge" werden vorgefertigte Textstücke und Sätze anhand von Regeln für Handlungsausgänge eingesetzt. "Black and White 2" ist nicht ganz so variabel und wählt nur vorgefertigte Texte

aus. Und auch viele Sportspiele, unter anderem "FIFA WM 2006", setzen vorgefertigte Texte anhand von Regeln für Spielsituationen zu Durchsagen des Stadionsprechers zusammen.
In seinem Vortrag unterscheidet Marcus Lechner simultane und zusammenfassende Textgenerierung. Bei Ersterer sollten geplante Handlungen des Spielers erahnt und nur auf Wissen, dass er schon besitzt, eingegangen werden. Bei Zweiterer müssen während des Spiels alle Handlungen und Ereignisse gespeichert werden und es können auch Informationen, die der Spieler nicht besitzt, mit einfließen.
Mögliche Kriterien, an denen sich die Generierung orientiert, sind die Rahmenhandlung des Spieles, vordefinierte Spielzustände, Handlungen des Spielers und der NPCs, vermutete Ziele, die sich ein Spieler selber setzt, Personen, die der Spieler trifft, Gegenstände, die er erhält, und der Verlauf von vorausgegangenen Spielen. Ein Problem stellt die Bewertung dieser Kriterien da. Sie kann statistisch geschehen, wobei sich beispielsweise die Frage stellt, ab wann eine Schlacht in einem Strategiespiel groß ist. Ab 20.000 oder erst ab 40.000 Einheiten? Um dieses zu umgehen, könnte man eine adaptive Bewertung verwenden, wobei vorangegangene Spiele als Referenz genommen werden. Eine Schlacht mit 100.000 Einheiten ist dann nicht groß, wenn es vorher schon einmal eine mit 200.000 Einheiten gab.
Um Texte zu generieren, beschreibt Marcus Lechner zwei Techniken. Einmal die Technik, die heute schon angewendet wird. Es existieren vordefinierte Satzstücke, in die Daten wie Volk, Spielername oder Datum eingesetzt werden, und Regeln, die angeben, wann die Satzstücke zu verwenden sind. Die zweite Technik ist, dass das System selbstständig natürliche Sprache generiert, wodurch keine Formulierungen vorab angegeben werden müssen und die KI leicht mit einem spezifischen Charakter versehen werden kann. Das kann unter Anderem durch Regelbasierte Systeme mit Fakten und Regeln, die die Prädikatenlogik verwenden, oder durch Syntax-Semantik-Verkettung realisiert werden. Dabei werden Fakten, was Spielerhandlungen seien können, in einzelnes Wissen zerlegt (das Faktum "Spieler läuft geradeaus" wird in das Wissen "Tom", den Spielernamen, und "zielstrebig" zerlegt). Zu diesen Wissensbausteinen gibt es eine Wortliste, die das Wissen ausdrückt. Zu den Worten ist die grammatikalische Eigenschaft (Substantiv, männlich etc.) und eine Liste von Synonymen und Antonymen[17] gespeichert, die statt dem Wort verwendet werden können. Anhand von grammatikalischen Eigenschaften der Sprache werden dann über Wahrscheinlichkeiten Sätze gebildet. Die heute existierenden Generatoren, die diese zweite Technik einsetzen, sind allerdings nach Angaben von Lechner für Spiele nicht praktikabel, weil ein benötigtes Wörterbuch einen Umfang von mehreren GB hat, die Satzgenerierung mehrere Minuten dauert und mindestens 30% der generierten Sätze inhaltlich oder grammatikalisch falsch sind.

[17] Ein Antonym ist ein gegensätzliches Wort. "Schwarz" ist ein Antonym zu "weiß".

# B Emotionen in der klassischen KI

## B.1 Einführung in die Emotionstheorie

### B.1.1 Emotionen beim Menschen

Für den Begriff der *Emotion* gibt es keine genaue Definition, die einheitlich benutzt wird. Aus diesem Grund verwende ich im folgenden eine Arbeitsdefiniton aus dem Fachbereich der Psycho-Physiologie. Emotionen sind psycho-physische Phänomene, die das Kernstück des Reiz-Evaluierungssystems des Gehirns bilden. Sie entstanden als evolutionäres Produkt, und bilden ein Signalsystem, das unsere Mitmenschen über unsere Gemütsverfassung und unsere Handlungsabsichten informiert[Are04]. *„Sie entstehen, ausgelöst durch Neurotransmitter, Neuropeptide und Hormone, vor allem im limbischen System, das Teile der Großhirnrinde(z.B des Stirnhirns) und viele Zentren im übrigen Gehirn umfasst“*[Mun01]. Das limbische System bewertet unsere Aktivitäten nach bestimmten Kriterien. Die Ergebnisse werden im emotionalen Gedächtnis abgespeichert. Vor der eigentlichen Speicherung einer neuen Information muss diese zuvor bewertet werden. Diese Bewertung findet statt, indem die Information mit bereites vorhandenem Wissen in Bezug gesetzt wird. Abhängig ist diese Bewertung ebenfalls vom derzeitigen emotionalen Zustand. Da Emotionen „biologische Reaktionen“ auf bereits vorhandene Inhalte sind, enthalten sie auch keine neuen Informationen. Sie sind lediglich eine subjektive Ausprägung des Menschen.

### B.1.2 OCC-Modell für Emotionen

Es existieren eine Vielzahl von Emotionsmodellen. Einen recht trivialen Ansatz verfolgt das Circumplex Modell, welches sogenannte Basisemotionen (z.B. Angst, Trauer, Wut, Furcht) in einem zweidimensionalen Koordinatensystem anordnet. Die x-Achse bezeichnet dabei die Valenz (hiermit wird angegeben, ob eine Emotion positiv oder negativ ist) und y-Achse die Aktivität (hiermit wird angegben, ob eine Emotion aktiv erlebt wird oder eher im Unterbewusstsein, also passiv, stattfindet). Dieses sehr einfache zweidimensionale Modell kann leicht auf ein dreidimensionales Modell erweitert werden, indem die dritte Achse die Intensität einer Emotion beschreibt. Ein einfaches Beispiel hierfür liefern verschiedene Intensitätsstufen für die Emotion *Angst*(dabei ist die Emotion *Panik* die höchste Ausprägung und die Emotion *Besorgnis* die niegste Ausprägung der Emotion *Angst.*). Einen anderen Ansatz verfolgt das OCC-Modell von Ortony, Clore und Collins aus dem Jahr 1988. Bei diesem Modell lassen sich aufgrund von verschiedenen Ausgangssituationen bestimmte Emotionen erzeugen. Das OCC-Modell unterscheidet zwischen ereignisfundierten, handlungsfundierten und objektfundierten Emotionen. Das Ziel ist es, menschliche Emotionen vorherzusagen und zu erklären. Bei ereignisfundierten Emotionen geht es um wertende Reaktionen auf Konsequenzen von Ereignissen (z.B. erfreut, nicht erfreut). Es wird zwischen Konsequenzen für andere und Konsequenzen für sich selbst unterschieden. Weiter lassen sich sowohl wünschenswerte bzw. nicht wünschenswerte als auch relevante bzw. nicht relevante Auswirkungen differenzieren. Unter handlungfundierten Emotionen versteht man ein kontrollierbares Herbeiführen oder Verhindern von Ereignissen durch eine Person(hier ist die eigene Person oder eine andere Person gemeint). Objektfundierte Emotionen beziehen sich auf Einzeldinge (z.B. Personen, Gegenstände oder Tiere). Insgesamt lassen sich auf diese Weise sechs situationsabhängige Emotions-Typen gewinnen (*fortunes-of-others*(Schicksal anderer), *well-being*(Wohlergehen), *prospect-based*(auswirkungsorientiert), *attribution*(Zuordnung), *well-being/attribution Mischformen*und *attraction*(Anziehung)) [Wal06]. Zuletzt gebe ich ein

abschließendes Beispiel an. Aus den in Tabelle 1.1 aufgeführten Informationen lässt sich

| |
|---|
| Ereignis(Strafzettel) |
| Konsequenzen für Andere |
| nicht wünschenswert für Andere |
| Feind |

**Tabelle B.1:** Beispiel

wahrscheinlich die Emotion *Schadenfreude* extrahieren.

## B.2 Wann gilt eine Maschine als intelligent?

### B.2.1 Der Turing-Test

Alan Mathison Turing (* am 23.06.1912 in London (England); † am 7.06.1954 in Wilmslow (England)) war ein bekannter Logiker, Mathematiker und Kryptoanalytiker. Ein großer Teil der theoretischen Grundlagen für die moderne Informations- und Computertechnologie beruhen auf seinen Ideen. Unter anderem gelang es ihm durch einfache, formale Geräte (sogenannten Turingmaschinen) die Unentscheidbarkeit des Entscheidungs- und Halteproblems zu zeigen. Turing schlug im Jahre 1950 in seinem erschienenen Artikel *Computing Machinery and Intelligence* [Tur50] einen Weg vor, um zu testen, ob eine Maschine intelligent ist oder nicht. Er bezeichnete diesen Test selbst als *the imitation game*. Zunächst ergibt sich für das Spiel ein Initialzustand an dem ein Mann, eine Frau und ein Befrager teilnehmen. Der Befrager darf sich dabei nicht im selben Raum auf wie der Mann und die Frau aufhalten. Der Befrager kann nur über ein Terminal(Tatatur und Bildschirm) mit dem Mann bzw. der Frau kommunizieren. Es besteht keine Möglichkeit, dass der Befrager andere Kommunikationkanäle nutzt, um mit seinem Gegenüber in Kontakt zu treten. Die Aufgabe des Befragers ist es, den beiden Personen Fragen zu stellen, welche aus beliebigen Fachgebieten stammen können, und anschließend festzusetzen, welche der beiden Personen die Frau ist. Die Aufgabe des Mannes bzw. der Frau ist es den Befrager zu überzeugen, dass die Person, mit der der Befrager gerade kommuniziert, die Frau ist.

Für Turing ergab sich nun die Fragestellung, was passieren wird, wenn eine Maschine die Rolle des Mannes übernimmt. Kann der Befrager so häufig wie zuvor erkennen, wer die Frau ist? Oder kann der Befrager überhaupt zwischen Mensch und Maschine unterscheiden? Falls der Befrager die Maschine nicht vom Menschen unterscheiden kann, dann darf die Maschine als intelligent gelten und hat den Turing-Test somit bestanden. Ein weiteres Modell liegt vor, wenn der Mann durch eine Maschine und die Frau durch einen Mann ersetzt wird, wobei sich die Frage ergibt, ob der Befrager immer noch die Frau erkennen kann, obwohl keine Frau mehr an dem Spiel teilnimmt. Kritisch an dem Frau-Mann-Befrager-Modell ist die Tatsache, dass es dem Mann erlaubt ist den Befrager zu verwirren, damit dieser eine falsche Identifikation durchführt. Ähnliches gilt für das Maschine-Mann-Befrager-Modell. Bei dem Maschine-Frau-Befrager-Modell kann die Frau dem Befrager unwahrheitsgemäße Antworten geben, um den Befrager auf die falsche Fährte zu locken. Der Turing-Test ist also irreführend, da es möglich ist, die Täuschung des Befragers als das Ziel des Turing-Testes mißzuverstehen.

Damit oben genannte Maschine menschenähnlicher wirkt, muss eine Abbildung menschlicher Emotionen auf sein Verhalten definiert werden. Dies kann nur gelingen, wenn die Maschine Kenntnisse über die menschliche Konstitution (mit dem Begriff der Konstitution ist die körperliche und seelische Verfassung eines Menschen gemeint) besitzt.

Beispielsweise muss die Maschine so programmiert werden, dass sie wie ein Mensch antwortet, was bedeutet, dass die Maschine verzögerte Antworten geben sollte, auch falsche Antworten zulässig sein sollten und ebenfalls Rechtschreib- und Flüchtigkeitsfehler machen sollte. Bereits hier zeigt sich wie wichtig Emotionen für Maschinen sein können, wenn es darum geht, eine Maschine mit menschlichen Eigenschaften zu konstruieren. In der heutigen Anwendung zeigen sich *Variationen* des Turing-Testes. Dabei werden die Leistungen eines intelligenten Programms für eine bestimmte Menge von Problemen auf funktionaler Ebene verglichen mit der Leistung eines menschlichen Experten auf diesem Gebiet.

### B.2.2 ELIZA

Joseph Weizenbaum (* am 8.01.1923 in Berlin (Deutschland)) ist ein deutsch-US-amerikanischer Informatiker sowie Wissenschafts- und Gesellschaftskritiker. Er entwickelte im Jahre 1966 am Massachusetts Institute of Technology (MIT) ein Computerprogramm namens ELIZA, mit dem es möglich war, eine Konversation zwischen Mensch und Maschine mittels Tastatur zu führen. Das Programm ELIZA wurde mit dem MAC Mehrbenutzersystem (engl. time-sharing system) realisiert, welches ein direkter Nachfolger des Compatible Time-Sharing System (CTSS) war. CTSS war eines der ersten Mehrbenutzersysteme und wurde ebenfalls am MIT entwickelt.

Bei dem Programm ELIZA handelt es sich um eine Konstruktion mit einer Zwei-Bänder-Anordnung, wobei sich auf dem ersten Band ein Sprach-Analysator und auf dem zweiten Band ein Skript befindet [Wei94]. Das Programm hat nun die Aufgabe Eingabestrings nach Schlüsselwörtern zu durchsuchen, wobei den Schlüsselwörtern Prioritäten zugeordnet werden. Der Sprach-Analysator auf dem ersten Band muss nun diese Schlüsselwörter erkennen. An diese Schlüsselwörter sind Regelsysteme zur Transformation gebunden, welche sich auf dem zweiten Band mit dem Skript befinden. Zusätzlich muss ein sogenannter „minimaler Kontext“ der Eingabesequenz gefunden werden. Aus dem minimalen Kontext und den Schlüsselwörtern bzw. mit diesen verbundenen Transformationsregeln generiert ELIZA eine neue Ausgabe. In dem Beispiel von Tabelle 2.1 ist

| | | | | | |
|---|---|---|---|---|---|
| Eingabe: | Hey, | you | are | stupid | ! |
| Ausgabe: | WHAT MAKES YOU THINK | I | AM | STUPID | ? |

**Tabelle B.2:** Beispiel

*“you“* das Schlüsselwort und *“you are“* der minimale Kontext der Eingabe. Daraus ergibt die oben abgebildete Ausgabe. ELIZA kann dabei nicht zwischen einer sinnvollen und sinnfreien Aussage unterscheiden. Man könnte das Wort *“stupid“* durch ein sinnloses Wort ersetzten und ELIZA würde den Unterschied nicht merken. Zum Schluss möchte ich folgende Sequenz einer Unterhaltung mit dem Computerprogramm ELIZA angeben [Wei66]:

Person: Men are all alike.
ELIZA: IN WHAT WAY
Person: They’re always bugging us about something or other.
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE
Person: Well, my boyfriend made me come here.
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE
Person: He says I’m depressed much of the time.
ELIZA: I AM SORRY TO HEAR YOU ARE DEPRESSED
Person: It’s true. I am unhappy

ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
Person: I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
Person: Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
Person: My mother takes care of me.
ELIZA: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
Person: My father.
ELIZA: YOUR FATHER
Person: You are like my father in some ways.
ELIZA: WHAT RESEMBLANCE DO YOU SEE
Person: You are not very aggressive but I think you don't want me to notice that.
ELIZA: WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE
Person: You don't argue with me.
ELIZA: WHY DO YOU THINK I DON'T ARGUE WITH YOU
Person: You are afraid of me.
ELIZA: DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU
Person: My father is afraid of everybody.
ELIZA: WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK OF YOUR FATHER
Person: Bullies.
ELIZA: DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

Die Antworten zeigen, dass jeder Input der Person zu einem Output von ELIZA führt, welcher in Form einer Frage ausgegeben wird. Bei einer solchen Implementierung ist es klar, dass es sich bei ELIZA nicht wirklich um ein intelligentes Programm handelt, da ELIZA Eingaben bzw. Ausgaben nicht versteht, sondern lediglich neue Fragen zurückgibt, die mit dem zuletzt eingegebenen Text zusammenhängen. In diesem Fall übernimmt das Programm die Rolle eines Psychaters. Bei der verwendeten Architektur ist es jedoch möglich, Skripte zu erweitern oder zu bearbeiten, was zu einer anderen Ausgabe führen wird und ein Gespräch mit einem anderen Gesprächspartner simuliert. Es ist bis heute noch nicht gelungen, Programme zu entwickeln, die einen *echten* Menschen simulieren können, jedoch führen die ständig zunehmende Rechnerperformance und Fortschritte im Bereich der Künstlichen Intelligenz zu besseren Resultaten.

## B.3 Emotionen in der klassischen KI

### B.3.1 Menschliche und Künstliche Intelligenz

Es gibt mehrere Definitionen für den Begriff der *Intelligenz* und den der *künstlichen Intelligenz* , daher ist es nicht möglich eine einheitliche Spezifikation für diesen Begriff anzugeben. Die folgenden Angaben zu diesem Abschnitt beschränken sich daher nur auf grundlegende Aspekte[1]. Künstliche Intelligenz(KI) ist ein Teilgebiet der Informatik, dennoch zählt sie nicht zu den klassischen Gebieten der Informatik, da der Begriff der Intelligenz immer auch individuelle Wahrnehmung, eigenständiges Denken und Handeln zur Folge hat, was symbolverarbeitende, logische Systeme in der Regel nicht können. Üblicherweise ist es so, dass einfachste Aufgaben, die ein junger Mensch mühelos bewältigen kann, wie zum Beispiel, zwischen einem Photo und einem realen

[1] Für detaillierte Informationen verweise ich daher auf entsprechende Fachliteratur wie [GRS03] oder [RN004].

Menschen zu unterscheiden, auf zwei Beinen zu gehen oder einen Weg vom Schlafzimmer zum Wohnzimmer zu finden, nicht als intelligente Leistungen gelten. Zu Beginn der 1980er Jahre war den meisten KI-Forschern jedoch klar, dass diese Probleme schwierig zu lösen waren. Sehen, gehen, navigieren, das erfordert gewöhnlich kein ausdrückliches Denken oder Ketten von Überlegungen, jedoch stellen diese Aufgaben eine Maschine vor große Probleme.

Die künstliche Intelligenz beschäftigt sich mit der Konstruktion informationsverarbeitender Systeme, welche kognitive, menschliche Fähigkeiten modellieren bzw. *intelligente* Leistungen erbringen sollen. Ein Ziel der künstlichen Intelligenz ist es Maschinen zu erstellen, die im Idealfall einem menschlichen Individuum in nichts nachstehen. Bereits heute sind grundlegende Erkenntnisse zur Verarbeitung von Informationen zwischen den Nervenzellen bekannt. Sie ähneln biochemischen Ja/Nein-Entscheidungen. Künstiche „Neuronale Netze“ können helfen, Software und Informationen aus vielen verschiedenen Datenbanken ständig neu miteinanander zu verknüpfen. Ein weiteres Ziel ist die Konstruktion von speziellen Anwendungen, für die auf den ersten Blick ein intelligentes Verhalten notwendig ist. Dazu gehören Erkennung und Analyse von Bildern, Erkennung und Synthese von Sprache, intelligente bzw. autonome Agenten, medizinische Diagnosesysteme und weitere. Die künstliche Intelligenz konnte sich in den letzten Jahren sehr stark weiterentwickeln, da auch bei der Erforschung der biologischen Grundlagen von intelligentem Verhalten bei Tieren und bei Menschen Fortschritte gemacht wurden.

### B.3.2 Emotionen in der Künstlichen Intelligenz

Nachdem der erste Abschnitt des ersten Kapitels sich mit einer Definition des Emotionsbegriffes beschäftigt hat und im ersten Abschnitt dieses Kapitels der Begriff der künstlichen Intelligenz angesprochen wurde, werde ich mich in diesem Abschnitt dem eigentlichen Thema dieser Arbeit, nämlich der Emotion in der klassischen KI, widmen. Erste Erkenntnisse wurden bereits im Jahre 1967 von Herbert Alexander Simon (* 15. Juni 1916 in Milwaukee (USA); † 9. Februar 2001 Pittsburgh (USA)) in seiner Publikation “Motivational and emotional controls of cognition“ veröffentlicht. Dabei geht es ihm darum, zu zeigen, wie emotionale Kontrollflüsse mit informationsverarbeitenden Systemen verbunden werden können, so dass die Datenverarbeitung in einer Vereinigung mit Emotionen und Gefühlen stattfindet. Er bezeichnet Emotionen als Interrupt-Systeme, welche bei eintreffenden Ereignissen mit hoher Proirität die derzeitige Aktivität blockieren können. Hierbei geht er von einem seriellen Betrieb des Interrupt-Systemes aus, wobei die Prioritäten ein hierarschiches Modell festlegen sollen [Sim67]. Ein aktuelles Beispiel gibt es im Bereich der Robotik mit dem am MIT entwickelten Roboter KISMET, der zur sozialen Interaktionen mit einem Menschen fähig ist [Are04]. KISMET's Systemarchitektur beinhaltet sechs Subsysteme, nämlich das *low-level feature extraction system* , das *high-level perception system* , das *attention system* , das *motivation system* , das *behavior system* und das *motor system* [Bre08]. Die Interaktion dieser Subsysteme führt zur Simulation von Emotionen mit Hilfe von verschiedenen Gesichtsausdrücken (Mimik wird durch Bewegung von KISMET's Ohren, Augenbrauen, Augenlidern, Lippen, Kiefer und Kopf hervorgerufen), Vokalaussprachen und Bewegungen. In Testreihen mit verschiedenen Probanden war es möglich, mit KISMET eine Unterhaltung zu führen, wobei der Roboter nur „Nonsens-Silben“ von sich gibt und nichts von dem Gesagten verstehen kann [Are04]. Die Frage, ob eine Maschine überhaupt Emotionen haben soll, hat in der Vergangeneit für viel Diskussionsbedarf gesorgt. In der Künstlichen Intelligenz verfolgte man lange Zeit das Konzept künstliche Systeme mit rationaler Intelligenz zu programmieren, was bei dem logikbasierten Berechnungsmodell von Maschinen am naheliegensten ist, jedoch wurden dabei alle anderen Formen von Intelligenz vernachlässigt. Dieser Ansatz war daher auch nur für künstliche Problemstellungen erfolgreich, wie zum Beispiel bei der Konstruktion von „Deep Blue“, einem hochentwickelten

Schachcomputer, der seinerseits im Jahre 1996 den amtierenden Weltmeister im Schach (Garri Kasparow) besiegen konnte. Andere Aufgaben konnte Deep Blue aber nicht bewerkstelligen. Ein Beispiel von vielen was letzendlich zu der Schlussfolgerung führte, *„dass die Definition von Intelligenz über das Rationale nur die höchste Ebene erfasst und andere, möglicherweise wichtige Bereiche wie zum Beispiel das Denken in Bildern oder die Beteiligung von Emotionen ausschließt, dass hier also fundamentale Aspekte fehlen.“*[Are04] Erschwerend kommt hinzu, dass zumindest eine Reihe von Emotionen ohne Bewusstsein nicht existieren können. Insbesonders sind dies alle Emotionen, die eine Konzeption vom *Selbst* voraussetzen, zum Beispiel Scham [Rue08]. Diese Tatsache macht die Konstruktion „emotionaler“ Systeme noch komplexer.

### B.3.3 Beispiel für ein emotionsbeschreibendes Regelsystem

Gemäß des OCC-Modells (Kapitel Eins, Abschnitt Zwei) lassen sich in der klassischen KI Regelsysteme in Abhängigkeit von Situationsbewertungen definieren, die emotionale Zustände beschreiben. Die Funktion $D(p, e, t)$ definiert die Wünschbarkeit (Desirability) von einer Person $p$ für ein Ereignis $e$ zur Zeit $t$. Sogennante *positive Konsequenzen* werden mit $D(p, e, t) > 0$ angegeben und *negative Konsequenzen* mit $D(p, e, t) < 0$. Die Funktion $I_g(p, e, t)$ beschreibt eine Kombination von globalen Intensitätsvariablen (z.B. Erwartbarkeit, Realität und Nähe des Ereignisses $e$ für Person $p$ zum Zeitpunkt $t$). $P_j(p, e, t)$ gibt das Potential zur Erzeugung eines Zustands der Freude $j$ (engl. joy) an. Wenn positive Konsequenzen vorliegen, lässt sich das Potential der Freude mit der Funktion $f_j$ wie folgt berechnen (Angabe in Pseudocode):

$$
\begin{aligned}
&if(D(p, e, t) > 0)\{\\
&P_j(p, e, t) = f_j(D(p, e, t), I_g(p, e, t))\\
&\}
\end{aligned}
$$

Bei negativen Konsequenzen, lässt sich entsprechend das Potential des Schmerzes $d$ (engl. distress) mit der Funktion $f_d$ berechnen:

$$
\begin{aligned}
&if(D(p, e, t) < 0)\{\\
&P_d(p, e, t) = f_d(D(p, e, t), I_g(p, e, t))\\
&\}
\end{aligned}
$$

Weiter lassen sich in Abhängigkeit von einem sogenannten Schwellwert $T$ Aktivierungsregeln für bestimmte Emotionen definieren (hier für die Emotion $Freude$ $(j)$):

$$
\begin{aligned}
&if(P_j(p, e, t) > T_j(p, t))\{\\
&I_j(p, e, t) = P_j(p, e, t) - T_j(p, t))\\
&\}else\ I_j(p, e, t) = 0
\end{aligned}
$$

Um die Intensität einer Emotion zu bestimmen müssen zunächst einige Begriffe definiert werden. $\epsilon_{pl}$ bezeichnet den Grad der Beeinflussung von Emotionen $\{p|p \in \{1, ..., P\}\}$ durch einen internen Stimulus $\{l|l \in \{1, ..., 4\}\}$, wobei die einzelnen natürlichen Zahlen für für $l$ vier verschiedene Stimuli (Neuronale Stimuli, Sensomotorische Stimuli, Motivationsstinuli und corticale Stimuli) bezeichnen [Mai99]. Hierbei bezeichnet ein Stimulus eine unwillkürliche Reaktion auf ein Ereignis, also einen auslösenden Reiz. Mit $\alpha_{pm}$ wird der Grad der Anregung und mit $\beta_{pm}$ der Grad der Hemmung von Emotion $p$ durch Emotion $m$ angegeben. Die Funktion $f$ berechnet die Intensität einer Emotion im Intervall der Schwellenwerte. Die Intensität einer Emotion ist beschränkt durch einen unteren Schwellenwert der Auslösung und einen oberen Schwellenwert der Sättigung. Mit der Funktion $g$ wird der Verfall eines emotionalen Potenzials im Laufe der Zeit beschrieben. Die Funktion $I_p(t)$ führt nun eine Berechnung der Intensität einer Emotion

$p$ in Abhängigkeit von der Zeit $t$ durch:

$$I_p(t) = g(f(I_p(t-1)) + \sum_{l=1}^{4} \epsilon_{pl} + \sum_{m=1}^{p} (\alpha_{pm} - \beta_{pm}) I_m(t))$$

Es ist klar, dass bei diesem regelbasierten Emotions-Modell Emotionen lediglich Resultate von bestimmten Situationen sind, wobei Emotionen ihrerseits ebenfalls bestimmte Situationen repräsentieren und auch neue Emotionen auslösen können. Das heißt, dass Emotionen beim verwendeten KI-System nicht erlebt werden, sondern in typischen Situationen ein typisches Verhalten zeigen. Allerdings ist das Erleben von Emotionen bei Maschinen nicht völlig ausgeschlossen, wenn die Software für die Emotionen mit der entsprechenden Wetware hormoneller, neurochemischer und physiologischer Abläufe verbunden wäre, jedoch steht dem eine hohe Komplexität entgegen [Mai99].

### B.3.4 Emotionsbezogene Forschungsgebiete in der Informatik

Grundsätzlich lässt sich die emotionsbezogene Forschung in der Informatik in drei Bereiche unterteilen. Der erste Bereich wird als "Emotion Recognition" bezeichnet. Hierbei soll der emotionale Zustand des Computernutzers anhand der Sprache, Gestik, Mimik, usw. erkannt werden. Um die Emotionen in der menschlichen Mimik festzulegen, haben die beiden Psychologen Paul Ekman und Wallace Friesen ein Klassifikationsschema der menschlichen Mimik eingeführt[2]. Um bei dieser Technik Informationen zu erhalten sind spezielle Features notwendig. Diese gewinnt man zum Beispiel aus Signalen(z.B. Audiosignale oder Videosignale) und muss anhand dieser erhaltenen Features entscheiden, welche Emotion vorliegt. Bei dem zweiten Gebiet der Emotionsforschung geht es um das sogenannte "Emotion Modeling". Dieses definiert ein Erklärungsmodell, welches für die Generierung von Emotionen in einem intelligenten System erstellt wird. Es ermöglicht einem virtuellen Agenten emotional zu handeln bzw. für den Betrachter so zu erscheinen [EF78]. Speziell in diesem Bereich kommen zuvor beschriebene regelbasierte Techniken zum Einsatz, die einen Agenten zur Erzeugung von emotionalen Handlungen motivieren. Bei der "Emotion Embodiment" werden emotionale Nachrichten, die zum Beispiel in Form von Mimik und Gestik visualisiert werden, von künstlichen Personen(Avataren) erzeugt. Interessant anzumerken ist, dass das wachsende Interesse an Emotionen in der KI-Forschung eine Parallele hat im wachsenden Interesse der Kognitionspsychologie an Emotionen. Neue Erkenntnisse in der Neurowissenschaft haben dazu beigetragen, dass dem emotionalen Subsystem eine weitaus höhere Bedeutung für das Funktionieren des menschlichen Geistes zuzuschreiben ist, als bislang angenommen [LeD96].

Für die Zukunft sind einige Einsatzgebiete denkbar. Ein Beispiel liefert eine automatische Produktbewertung, bei der Testbenutzer während der Testphase eines Produktes von Prototypen beobachtet werden und diese Protypen die Reaktionen der Testbenutzer aufnehmen und auswerten. Ein weiteres Beispiel ergibt sich beim Datenaustausch im Internet. Hier kann bei der Kommunikation in einem Chat die Mimik des Benutzers analysiert und kodiert werden. Auf der Gegenseite wird sie durch eine künstliche Person dargestellt. So wird die Anonymität gewahrt, die Kommunikation jedoch angereichert. Bei der Konstrukion eines elektronischen Vorlesers könnten entsprechende Stellen im Text mit der adäquaten Betonung vorgelesen werden. Es sind noch viele weitere Beispiele vorstellbar [Rei08].

[2] Hierbei handelt es sich um die beiden Techniken FAST(Facial Affect Scoring Technique) und Facial Action Coding System(FACS) [EF78].

# C Evolutionäre Algoithmen

## C.1 Einleitung

Evolutionäre Algorithmen (EA) sind radomisierte Suchheuristiken, welche biologische Evolutionsmechanismen verwenden. Diese EA wurden schon auf schwierige Probleme angewendet und haben sehr gute Ergebnisse erzielt.
Eines der ersten Beispiele ist die Optimierung einer Zweiphasen-Überschalldüse, welche von Schwefel 1968 durchgeführt wurde. Dabei wurden mehrere Prototypen durch zufällige Kombination von Sektionen bereits gebauter Düsen hergestellt und getestet. Mit diesem Vorgehen konnte eine Verbesserung des Wirkungsgrades um 40% erziehlt werden [Sch68].
Bei vielen Problemen sind physische Prototypen aber nicht nötig, da ein Computer die Qualität einer gefundenen Lösung berechnen kann. Im Folgenden wird behandelt, wie ein EA aufgebaut werden kann.

## C.2 Grundbegriffe

Zunächst werden einige Begriffe eingeführt, welche im Zusammenhang mit Evolution und EA stehen.
In der Biologie stellt der Begriff „Population“ eine Gruppe von gleichartigen Individuen dar, welche eine Fortpflanzungsgemeinschaft bilden. Wenn dies auf die Informatik übertragen werden soll, fehlen nur die Definitionen für „Individuum“ und „Fortpflanzung“.

### C.2.1 Suchraum

Ein Individuum repräsentiert bei einem EA einen möglichen Lösungskandidaten im Suchraum $S$, welcher $\mathbb{R}^n$, $\mathbb{B}^n$, $S_n$, der Raum der Bäume und Graphen oder ein problemspezifisches Konstrukt sein kann. Damit der EA möglichst zeit- und platzeffizient arbeiten kann, wird häufig der Suchraum kodiert. Der kodierte Suchraum wird Genotyp bezeichnet, während der Raum der konkreten Lösungen, also ihrer Bedeutungen, als Phänotyp bezeichnet wird.
Als Beispiel sei das Problem des Handlungsreisenden betrachtet. Ein Individuum wäre in dem konkreten Beispiel eine Tour in dem Suchraum aller möglichen Touren. Ein mögliches Individuum könnte zum Beispiel im Phänotyp „Gehe von Dortmund(2) über Bochum(3) und Essen(1) nach Duisburg(4) und kehre anschließend nach Dortmund(2) zurück“ lauten und durch das Genom (eine Ausprägung des Genotyps) 2314 dargestellt werden.

### C.2.2 Gray-Kodierung

Binäre Kodierungen können dem evolutionären Prinzip entgegenwirken, welches bei kleinen Veränderungen im Individuum auch kleine Änderungen in der Lösung erwartet. Sucht man beispielsweise einen optimalen Parameter $\omega \in \mathbb{N}$, würde der Wert 42 in Standard-Bitkodierung 101010 entsprechen. Invertiert man jetzt eine Eins, kann man die Lösung 100010 erhalten, was der Dezimalzahl 34 entspricht. Eine Verbesserung bietet die Gray-Kodierung [Bae96, S. 221 ff.] [Hol71, Bet80]:

**Tabelle C.1:** Gray- und Standardbinarkodierung

| Zahl | stdbin-kod* | Gray | (* Standardbinärkodierung) |
|---|---|---|---|
| 0 | 000 | 000 | |
| 1 | 001 | 001 | |
| 2 | 010 | 011 | |
| 3 | 011 | 010 | |
| 4 | 100 | 110 | |
| 5 | 101 | 111 | |
| 6 | 110 | 101 | |
| 7 | 111 | 100 | |

Wie man anhand der Tabelle sehen kann, beträgt der Hammingabstand ($d_{hem}(A,B) = |\{i \in \mathbb{N}^{\leq n} | A_i \neq B_i \wedge A, B \in \mathbb{B}^n\}|$) zweier graykodierter Bitmuster, welche im Dezimalsystem aufeinanderfolgen, nie mehr als 1.
Um ein Standardbinärmuster $B = (b_1, \ldots, b_n) \in \mathbb{B}^n$ in die Graykodierung zu überführen, wird auf alle Bits $b_i \in B$ die folgende Transformation angewendet ($\otimes$ sei als das *exklusive Oder* definiert):

$$g_i = \begin{cases} b_i & i = 1 \\ b_{i-1} \otimes b_i & sonst \end{cases}$$

Für die Rücktransformation gilt:

$$b_i = \begin{cases} g_i & i = 1 \\ \bigotimes_{j=1}^{i} g_j & sonst \end{cases}$$

Man sieht allerdings an der Tabelle, dass dieses Verfahren keine Garantie bietet, dass das oben erwähnte Problem nicht auftritt. $2 \widehat{=} 011$ wird mit Invertieren der ersten 0 zu $111 \widehat{=} 5$
Alternativ könnte man auch nur Lösungen zulassen, die einen Hammingabstand fixer Größe nicht überschreiten bzw. eine Form der Variation wählen, welche dies gewährleistet.

### C.2.3 Zielfunktion

Bei jedem Optimierungsverfahren muss festgelegt werden in welche „Richtung" optimiert werden soll bzw. welche Lösung besser ist als eine andere. Dafür ist dann eine sogenannte Zielfunktion $f$ notwendig, die einem Individuum möglichst schnell einen numerischen Wert ($\mathbb{R}$ oder eine Teilmenge davon) zuweist.

$$f : S \to \mathbb{R}$$

Bei Zielfunktionen ist es von Vorteil, wenn man bestimmte Vorkenntnisse über den Lösungsraum hineinkodiert. Dabei darf man sich aber nicht von „Vorurteilen" beirren lassen. Reine Vermutungen können, da sie den Suchraum möglicherweise stark beschneiden, zu suboptimalen Ergebnissen führen. Zudem ist noch wichtig festzulegen, ob der Zielfunktionswert minimiert oder maximiert werden soll. In allen nachfolgenden Beispielen sei $f$ die Zielfunktion, die zu maximieren ist.

### C.2.4 Nebenbedingungen

In den meisten Fällen lassen sich Lösungswerte nicht beliebig maximieren, da sie an Restriktionen gekoppelt sind. Beispielsweise versucht man beim Problem *Vertex Cover* in einem Graph $G = (V, E)$ eine minimale Knotenauswahl $V' \subseteq V$ zu finden, so

dass die gesammte Kantenmenge $E$ mit einem Knoten aus $V'$ inzident ist. Kodiert man das Problem in einen Bitvektor der Länge $|V|$, so belohnt man selbstverständlich die Individuen, welche weniger Knoten mit dem Binärwert 1 markieren. Unter diesen Voraussetzungen würde der EA immer eine geringe Knotenauswahl treffen, welche aber die Eigenschaften eines *Vertex Covers* nicht erfüllt, dass nämlich alle Kanten mit Knoten des Covers inzident sind.
Eine Möglichkeit, dieses Problem zu lösen, ist die Einführung eines Strafterms in der Zielfunktion, welcher in Abhängigkeit der Verletzung der Nebenbedingungen den Funktionswert verändert. In diesem Beispiel würde sich die Anzahl an nicht abgedeckter Kanten zum Quadrat als Strafterm anbieten, welches das Auslassen von großen Knotenmengen verbieten würde. Andererseits ist das Auslassen weniger Kanten immer möglich.
Eine andere Möglichkeit der Berücksichtigung von Nebenbedingung ist die Barriere-Methode. Dabei wird der Suchraum durch eine definierte Barriere beschränkt, so dass ein Annähern an diese Barriere mit asymptotisch unendlich anwachsenden Kosten bestraft wird. Dadurch wird ein Verlassen des durch die Nebenbedingungen beschränkten Suchraums unmöglich. In [Hor79, Kap. 3.5.1 f.] geht Horst detailierter auf beide Techniken ein.

### C.2.5 Variation

Als nächstes ist der Begriff der „Fortpflanzung“ zu klären. In einem EA kann dafür die *Mutation* verwendet werden. Bei diesem Verfahren werden bei einer Kopie eines Elternindividuums einzelne Merkmale verändert in der Hoffnung auf eine bessere Anpassung an die Umwelt (d.h. einem besserem Ergebnis beim Optimierungsproblem). Daneben gibt es noch die die *Rekombination* (auch *Crossover* genannt). Dabei wird aus den Merkmalen zweier oder mehrerer Individuen ein Nachkomme erzeugt, und man hofft, dass sich die guten Merkmale der Eltern durchsetzen oder sogar gegenseitig verstärken. Häufig wird nach einer Rekombination noch eine Mutation auf den Nachkommen angewendet.
Damit sich in einer Population gute Individuen entwickeln können, müssen in jeder Generation die Anzahl an Elternindividuen ($\mu$) und die Anzahl der von den Eltern abstammenden Kindern ($\lambda$) festgelegt werden.

### C.2.6 EA-Schema

Mit diesem groben Wissen über die einzelnen Module lässt sich nun ein allgemeines Aufbauschema eines EA formulieren.

- *Initialisierung* einer Population
- *Evaluierung* der Individuen mit Zielfunktion

  solange *Abbruchkriterium* nicht erfüllt ist, starte Generationszyklus:

  - ◇ *Selektion* zur *Reproduktion* (Elternindividuenauswahl)
  - ◇ *Reproduktion* durch Variation (Mutation/Rekombination)
  - ◇ *Evaluierung* der Kinder
  - ◇ *Selektion* der *Überlebenden* (Schrumpfen der Population)
- *Ausgabe* des besten Individuums

Mögliche Abbruchkriterien können beispielsweise beschränkte Ressourcen sein (Beschränkung der Generationszyklen), qualitative Kriterien (Erreichen von Wunschqualitäten

der Lösungen, Konvergenz [$k$ Generationen keine Verbesserung]) oder externe Abbruchsignale (z.B. Ungeduld des Benutzers).

## C.3 EA-Operatoren

### C.3.1 Selektion

Bei EA spielen Selektionsoperatoren für die Population eine große Rolle. Die Frage welche Individuen in einer Population bleiben sollen bzw. mit welchen neue Kinder generiert werden sollen, wird in diesem Abschnitt näher thematisiert.
Zuvor muss man aber noch festlegen, aus welcher Menge von Individuen die Eltern selektiert werden soll. Es gibt zwei klassische Varianten.
Die $(\mu, \lambda)$-EA selektieren *nur* aus den $\lambda$ Kindern die neuen $\mu$ „Erzeuger" der nächsten Generation, während die $(\mu + \lambda)$-EA aus Kindern *und* Eltern die neuen $\mu$ „Erzeuger" der nächsten Generation selektieren. Ein $(\mu + 1)$-EA wird in der Literatur auch als Steady-State-EA bezeichnet.
Es gibt noch die Möglichkeit einem Individuum eine „Lebensdauer" zu geben, indem man die Notation $(\mu, \kappa, \lambda, \rho)$ verwendet. Hierbei „überlebt" ein Individuum nur $\kappa$ Generationszyklen. $\rho$ beschreibt hier das Variationsverfahren ($\rho = 1$ : Mutation, $\rho > 1$: Rekombination mit $\rho$ Eltern) [SR95].

#### Besten-Selektion

Von den Selektionsoperationen ist die einfachste - und aus Darwinistischer Sicht auch die einleuchtendste - Methode die Besten-Selektion (auch *survival-of-the-fittest* genannt). Dabei werden die besten Individuen, gemäß der durch die Zielfunktion bestimmten Fitness, für die nächste Generation ausgesucht.

#### q-stufige Turnierselektion

Eine etwas „großzügigere" Art der Selektion (gegenüber schwächeren Individuen) stellen die probabilistischen Selektionsmechanismen dar. Da die Selektion zufallsgesteuert ist, haben Individuen mit einem niedrigeren Zielfunktionswert bessere Chancen ausgewählt zu werden als bei der Besten-Selektion.
Bei der *q-stufigen Turnierselektion* (eingeführt in [Fog95]) wird für jedes Individuum gleichverteilt q(>1) Gegner ausgelost, gegen die es verglichen wird. Die Anzahl der Siege (=Anzahl an Vergleichen mit anderen $q$ Individuen, bei denen der Zielfunktionswert echt besser war) ergibt eine Rangreihenfolge für alle Teilnehmer. Gemäß dieser Reihenfolge werden die besten Individuen ausgewählt.
So wie bei der Besten-Selektion können bei diesem Verfahren keine besten Individuen verloren gehen. Ein Individuum kann nur bei Kämpfen gegen gleichstarke oder stärkere Gegner im Rang geschwächt werden.

Im folgenden Beispiel werden 2 Individuen aus einer Population der Kardinalität 4 durch eine 2-stufige Turnierselektion ausgewählt.

#### Fitnessproportionale Selektion

Ein weiterer probabilistischer Selektionsansatz ist die fitnessproportionale Auswahl. Der Grundgedanke dabei ist, dass ein höherer Fitnesswert nur die Wahrscheinlichkeit verschiebt, wie häufig das Individuum selektiert wird. Wenn man sich die Selektion als

**Tabelle C.2:** Beispiel: 2-stufige Turnierselektion mit Population der Größe 4

| Individuum | Fitness | Gegner | | | | #Siege | selektiert? |
|---|---|---|---|---|---|---|---|
| (1) | 3.2 | (3) | ✓ | (2) | | 1 | ✓ |
| (2) | 4.6 | (4) | | (1) | ✓ | 1 | |
| (3) | 1.2 | (2) | | (4) | | 0 | |
| (4) | 6.3 | (1) | ✓ | (3) | ✓ | 2 | ✓ |

Roulettespiel vorstellt, dann sind auf dem Roulette-Rad alle Individuen durch Felder vertreten. Die Größe der Felder richtet sich nach der relativen Fitness des jeweiligen Individuums.
Die Wahrscheinlichkeit dass ein Individuum $i$ gewählt wird beträgt:

$$Prob(x_i) = \frac{f(i)}{\Sigma_{j=1}^{n} f(j)}$$

Das folgende Beispiel zeigt die Selektion der einer Population, deren Fitnesswerte durch das Tupel $(2, 4, 4, 5, 9)$ repräsentiert wird.

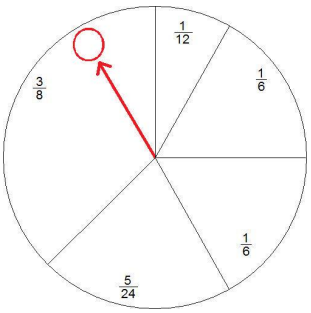



**Abbildung 39:** Fitnessproportionales Roulettespiel mit 5 Individuen

Möchte man in einer Population, welche die Größe $r$ hat, $s$ Individuen selektieren, muss man pro Selektion (welche $O(s)$ mal durchgeführt wird) das Feld „suchen“, welches der Wahrscheinlichkeit der Kugel entspricht. Mit binärer Suche dauert das $O(log(r))$. Mit der Transformation der Fitnesswerte in Wahrscheinlichkeiten ($O(r)$) ergibt sich daraus eine Gesammtlaufzeit von $O(r + s \cdot \log(r))$.

### Stochastisch universelles Sampling

Will man bei dem Selektionsvorbild des fitnessproportionalen Roulette-Spiels bleiben, wäre das *stochastisch universelle Sampling* eine Alternative zum vorherigen Verfahren. Das Prinzip ist hier gleich geblieben, da man wieder ein Roulette-Rad erstellt, bei dem Größe der Felder den relativen Fitnesswerten der Individuen entspricht.
Die eigentliche Selektion, welche bezogen auf Roulette dem „Werfen“ der Kugel auf das (sich drehende) Rouletterad entspricht, unterscheidet sich zur fitnessproportionalen Selektion dadurch, dass (anstelle von einer Kugel) $s$ Kugeln verwendet werden, die alle den identischen Abstand zueinander haben. Dieses Verfahren wird in [Bak87] ausführlicher behandelt.
Als Beispiel wird die Population aus dem vorherigen Abschnitt genommen. Hierbei seien $r = 5$ und $s = 3$ (Abb. 40)

Die Transformation der Fitnesswerte in Wahrscheinlichkeiten dauert hier wieder $O(r)$. Das „Werfen“ der Kugeln wird einmal getätigt. Für das „Suchen“ der Individuen, welche von den Kugeln ausgewählt wurden, braucht man nur $O(r + s)$. Die Laufzeit kann man

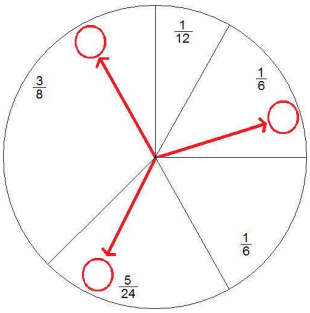



**Abbildung 40:** Stochastisch universelles Sampling am Rouletterad mit $r = 5$ und $s = 3$

damit erklären, dass einmal entlang des ganzen Rouletterades nach den selektierten Feldern gesucht wird ($O(r)$). Da der Parameter für die Größe der auszuwählende Multimenge $s$ auch größer als $r$ sein kann, kann sich die Laufzeit für die Suche höchstens um $O(s)$ verlängern. Daraus ergibt sich eine Gesamtlaufzeit von $O(r + s)$.

### C.3.2 Mutation

Wie zuvor erwähnt, verändert man mit der Mutation einzelne Lösungen in der Hoffnung, bessere Individuen zu erhalten. Dabei ist die Mutationshäufigkeit eines einzelnen Gens in der Regel von einer Mutationsrate abhängig.
Im Abschnitt C.2.2 wurde die Einführung der Gray-Kodierung damit begründet, dass kleine Veränderungen in der Repräsentation kleine Veränderung in der Bedeutung mit sich ziehen sollen. Analog zu dieser Aussage ist es für einen Mutationsoperator (dessen Arbeitsweise von Wahrscheinlichkeiten beeinflusst wird) im Allgemeinen wünschenswert, dass Veränderung, welche den gleichen „Aufwand“ haben, auch die gleiche Wahrscheinlichkeit der Durchführung haben. Dieser Aufwand ist messbar, wenn man eine Abstandsfunktion für den Suchraum hat. Im Raum $\mathbb{B}^n$ erfüllt das der Hammingabstand (siehe C.2.2) und im Raum $\mathbb{R}^n$ der euklidische Abstand.
Bei einem Elterindividuum $(1, 1, 0)$ aus $\mathbb{B}^3$ sollten demnach die Wahrscheinlichkeit die Kinderindividuen $(1, 0, 0)$ und $(1, 1, 1)$ zu erzeugen gleich sein, da sie jeweils zum Elterindividuum den Hammingabstand 1 haben. Das Kind $(0, 0, 1)$ zu erzeugen sollte dann folglich unwahrscheinlicher sein, da zwischen Kind und Elter ein Hammingabstand von 3 ist.
Analog zu diesen Wünschen an einen Mutationsoperator lassen sich auch Wünsche an einen Rekombinationsoperator (Kap. C.3.3) formulieren.
Für alle wahrscheinlichen Kinderindividuen soll gelten, dass das Maximum der Abstände zu deren Eltern durch die Entfernung der Eltern untereinander beschränkt ist. Zudem ist die Forderung sinnvoll, dass die Abstände zwischen dem Kind und all seinen Eltern gleichwahrscheinlich sind.

#### Standardbitmutation

Von den Mutationsverfahren, die unter anderem für den Suchraum $\mathbb{B}^n$ geeignet sind, ist die binäre Mutation eine der einfachsten. Dabei wird pro Bit des Elterngenstrangs eine Zufallszahl $u$ aus dem Intervall $[0, 1)$ gezogen. Abhängig von $u$ und der *Mutationswahrscheinlichkeit* wird das Bit invertiert oder nicht. Wünscht man, dass durchschnittlich nur ein Bit mutiert wird, eignet sich die Mutationswahrscheinlichkeit $p_m = \frac{1}{n}$ [Mue92] [Bae96, S. 229, 254].

### Gauss-Mutation

Für den Suchraum $\mathbb{R}^n$ eignet sich die Gauss-Mutation. Sie verwendet die Gaussfunktion $\mathcal{N}(0, \sigma)$ als Quelle von normalverteilten Zufallszahlen, deren Werte zu den zu verändernden Genen addiert wird. Die Dichtefunktion dieser Verteilung beträgt:

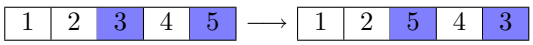
$$\phi(x) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} \cdot \exp\left(\frac{-1}{2 \cdot \sigma^2} \cdot x^2\right)$$

$\sigma$ stellt bei der Funktion die Standardabweichung dar. Übertragen auf das mutierte Individuum definiert es die „Schrittweite“, mit der es sich verändert. Zusätzlich zu der Schrittweite kann noch ein Wertebereich angegeben werden, den die einzelnen veränderten Gene nicht verlassen dürfen. Nähere Untersuchungen sind in [Rec73] zu finden.

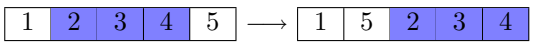
### Mutation von Permutationen

Die bisherigen Mutationverfahren eignen sich nicht für Permutationen, da jede Operation auf ein Gen unabhängig von den anderen Genen stattfand. Beim Individuum $(1, 2, 3) \in S_3$ würde die Mutation zu $(1, 3, 3)$ keine gültige Permutation hervorbringen. Daher werden in diesem Abschnitt passende Mutationen vorgestellt.

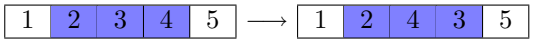
Der *Zweiertausch* tauscht zwei zufällig gleichverteile Elemente einer Permutation aus.



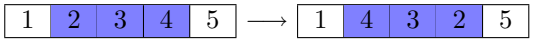

Beim *Verschieben eines Sequenz-Teilstückes* werden zwei Punkte $(a, b \in \{1, \ldots, n\}) \wedge a \leq b)$ auf dem Genom gewählt und anschließend um $\lfloor u \cdot l \rfloor$ zirkulär im Genstrang verschoben, wobei $u \in [0, 1)$ zufällig gleichverteilt gewählt ist und $l = n + a - b$:




Beim *scramble sublist*-Operator werden zwei Punkte ausgewählt, zwischen denen die Elemente stochastisch permutiert werden [Dav91, S. 81 f.]:




Bei der *Inversion* werden die Elemente zwischen 2 Punkten umgedreht:




## C.3.3 Rekombination

### k-Punkt Rekombination

Eine Möglichkeit, neue Kinder durch Rekombination zu bilden, stellt die $k$-Punkt Rekombination dar. Hierbei werden zufällig $k$ Positionen über den Genotyp verteilt. Dadurch entsteht für den EA ein Muster, das definiert, wo er von dem einen Elternteil zum anderen springen muss, während er die Informationen der Eltern in jeweils ein Kind überträgt. Diese Rekombination ist ein Beispiel für kombinierende Rekombination.
Als Beispiel werden die beiden binärcodierten Eltern rekombiniert. Die $k = 2$ Stellen seien 2 und 5:

### Uniformer Rekombination

Eine weitere Möglichkeit stellt die *uniforme Rekombination* dar. Hier wird für jedes Genpaar der Eltern ein Zufallsbit gezogen, das entscheidet, von welchem Elternteil die Information in das Kind einfügt werden. Das andere Kind bekommt dann die Komplementärauswahl. Das Beispiel (Abb. 42) zeigt 2 Elternindividuen und eine Ansammlung an Zufallsbits in der Länge eines Individuums. Auch dieser Operator erzeugt kombinierend, wie bei der $k$-Punkt Rekombination, aus den Eltern neue Nachkommen.

### Arithmetische Rekombination

Die arithmetische Rekombination ist ein Operator, der sich nur bei reellwertigen Genen eignet, da sie auf der Idee beruht, Individuen implizit in einem Vektorraum abzubilden und Kinder darin zu suchen. Dabei kann für keinen Nachkommen garantiert werden, dass dieser aus ganzzahligen oder binären Genen besteht.
In seiner interpolierenden Funktion wird ein Nachkomme auf der Strecke zwischen seinen beiden Eltern gebildet (Abb. 43). Für alle Gene $A_i$ und $B_i$ der Elternindividuen wird genau *eine* Zahl $u$ zufällig gleichverteilt aus $[0, 1)$ gezogen. Dann gilt für alle Kindgene $C_i = u \cdot A_i + (1 - u) \cdot B_i$ [Mic92].
Wright [**?**] schlägt noch ein weiteres arthmetisches Rekombinationsverfahren vor.

### Uniform Order Based Crossover

Entspricht der Genotyp dem Raum aller Permutationen, so kann man nicht einfach die Gene zweier Eltern mischen, da bei den bisherigen Verfahren nicht sichergestellt ist, dass wieder eine Permutation entsteht. Der *uniform order based crossover* (vorgestellt in [Dav91, S. 72-90]) ermöglicht dies, indem er neben zwei Eltern auch eine gleichverteilte Zufallsbitmaske der Länge eines Individuums benutzt.
In einer Zwischenstufe werden alle Gene des ersten Elternteils an den Stellen des Wertes 0 der Bitmaske in das erste Kind eingefügt, analog alle Gene des zweiten Elter an den Stellen des Wertes 1 der Bitmaske. Die fehlenden Werte des Kindindividuums werden in der Reihenfolge des jeweils anderen Elternteils ergänzt (Abb. 44).

## C.4 Parametereinstellungen

Bei der Modellierung eines EA ist die Wahl von Parametern entscheidend für das Vorgehen der Operatoren. Am Beispiel von Mutationsoperatoren lässt sich mit Parametern einstellen, wie stark sich das Kindindividuum vom Elter unterscheiden soll. Im folgenden sei der zu betrachtende Parameter $\psi$.
Neben der *statischen* Parameterfestlegung gibt es noch die *dynamische* Anpassung [KDP83]. Dabei wird der Parameter nach einer gewissen Zeit angepasst. Zum Beispiel kann die $\psi$ alle zwei Generationen mit dem Faktor 0.98 verringert werden. Eine Verallgemeinerung dieses Vorgehens ist die *adaptive* Anpassung. Hierbei ist die Veränderung eines Parameters $\psi$ von einer Erfolgsrate, also den Verbesserungen in den letzten $k$ Generationen, abhängig. Um diese Erfolgsrate bewerten zu können, wird ein Schwellenwert $\Theta$ verwendet. Abhängig davon ob die Erfolgsrate bezüglich $\Theta$ größer, kleiner oder gleich ist, wird $\psi$ verändert. Eine spezielle Art der adaptiven Anpassung ist die $\frac{1}{5}$-*Erfolgsregel* (ausführlicher behandelt in [Rec73]).
Eine weitere Technik ist die *selbstadaptive* Anpassung, bei der zu jedem Individuum gespeichert wird, mit welchem Parameterwert von $\psi$ es gebildet wurde. Wenn das Individuum dann selber Nachkommen bilden soll, wird eine Evolutionsstrategie auf den gespeicherten Parameter angewendet und mit dem variierten Wert $\psi'$ ein oder mehrere

Nachkommen gebildet. Dadurch hofft man, dass „gute“ Parameterwerte „gute“ Individuen erzeugen und bei der *Selektion der Überlebenden* auf den Individuen erhalten bleiben. Diese Technik geht auf eine Arbeit von Schwefel [Sch77] zurück.

## C.5 EA in Spielen

Da man EA auf sehr viele Problemstellungen anwenden kann, stellt sich die Frage, ob sie auch für Spiele geeignet sind. Ponsen [Pon04] hat einen Warcraft2-Mod (Wargus), welcher die Open-Source-Strategie-Engine Stratagus benutzt, verwendet, um regelbasierte NPC-KI zu implementieren, zu testen und mit einem EA neue Offlinestrategien zu finden.
Das Ziel des EA war es, gegen zwei statische Strategien (Soldier's Rush & Knight's Rush) Gegenstrategien zu finden. Ein Individuum ist ein Regelwerk, welches aus Zuständen besteht. Jeder Zustand beinhaltet Regeln, welche für verschiedene Aspekte des Spieles (Gebäude bauen, Technologien erforschen, Wirtschaft und Kampf) zuständig sind. Der Algorithmus sollte terminieren, wenn die gefundene Strategie die gewünschte Qualität hatte oder wenn eine maximale Anzahl an Individuen erzeugt wurde. Ersteres war der Fall und die gefundenen Strategien zeigten ein sehr gutes durchschnittliches Verhalten bei den Evaluierung per Spielsimulation. Durch diese Ergebnisse war Ponsen motiviert, weiter in dieser Richtung zu forschen [PS04, PMASA05, PMAPA06].

## C.6 Wann werden EA eingesetzt?

Abschließend wird die Frage untersucht, in welchen Fällen man auf evolutionsbasierte Verfahren zurückgreifen sollte, wann sie sinnvoll sind und wann eher unsinnig. EA haben den Vorteil, dass sie relativ einfach zu implementieren sind und bei geeigneten problemspezifischen Anpassungen sehr gute Ergebnisse liefern können. Jedoch fallen EA unter den Begriff „Heuristik“, was bedeutet, dass man keine Qualitätsaussagen über die Lösung treffen kann. Man kann also weder sagen, ob die Lösung optimal ist, noch wie weit diese vom Optimum entfernt ist. Daher sollte man nicht auf EA zurückgreifen, wenn es für das Problem einen effizienten polynomiellen Algorithmus gibt, der eine optimale Lösung liefert.
Sie stellen aber gegenüber Rechenverfahren, die einen sehr hohen Aufwand an Rechenzeit benötigen, eine gute Alternative dar. Allerdings darf der Ressourcenaufwand, den EA benötigen, nicht unterschätzt werden. Dieser ist vom Entwickler skalierbar, da er dem EA beliebig viel Speicherplatz zusichern kann und ihn somit auch mächtiger macht, jedoch auf Kosten des Rechenaufwands für die Verwaltung des Platzes.
Die im Verlauf des Textes vorgestellten Operatoren und Verfahren bieten eine Fülle an möglichen Gestaltungsvariationen für EA, so dass es für unerfahrene Entwickler schwierig sein kann, einen effizienten Algorithmus zu modellieren. Umgekehrt gilt aber auch, dass durch dieses breite Arsenal EA auf sehr viele Probleme anwendbar sind. Hier sei auf eine etablierte Implementierung eines EA hingewiesen (CMA-ES [KMH$^+$04, Han07]).
Da es grundsätzlich keinen EA gibt, der bei beliebigen Problemen gleichgut arbeitet, bedarf es häufig problemspezifischer Anpassungen an den Algorithmus, um ihn effizient zu gestalten [MdWS91]. Auch wenn ein EA aus Qualitätsgründen nicht auf jedes Problem angewendet werden sollte, lässt sich für jedes Optimierungsproblem eine Zielfunktion modellieren, mit der der Algorithmus dann zufriedenstellende Ergebnisse liefern kann. Das machen EA so vielseitig einsetzbar bei beliebigen objektiv bewertbaren Optimierungsproblemen[Jan05, Wei07].

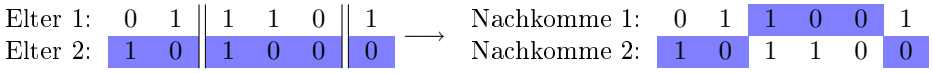



**Abbildung 41:** 2-punkt-Rekombination

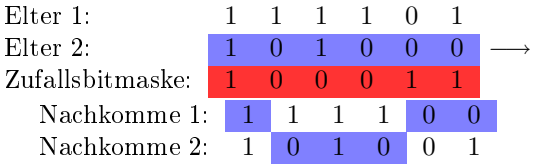



**Abbildung 42:** Beispiel einer uniformen Rekombination

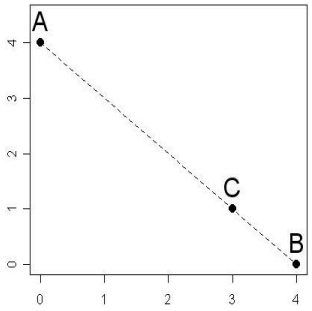



**Abbildung 43:** Beispiel einer artihmetischen Rekombination im $\mathbb{R}^2$

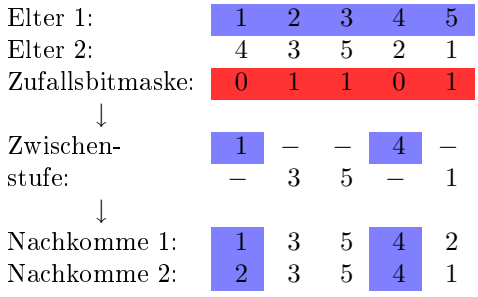



**Abbildung 44:** Beispiel eines uniform order based crossovers

# D Fuzzy-Systeme

Diese Ausarbeitung dient der Einführung in die Grundlagen der Fuzzylogik und der Fuzzy-Systeme. Zusammen mit neuronalen Netzen werden den Fuzzy-Systemen, aufgrund zahlreicher erfolgreicher Verwirklichungen in der Praxis, auch zukünftig hohes Potenzial für eine Erweiterung auf neue Anwendungsbereiche zugeordnet. Insbesondere der Bereich der Neuro-Fuzzy-Systeme, der versucht die Vorteile der beiden Verfahren zu vereinigen, erlangt in der Wissenschaft immer mehr Aufmerksamkeit und Popularität. Die Verwendung von Fuzzy-Systemen bewährt sich besonders, falls das System aufgrund seiner Komplexität nicht exakt beschrieben werden kann.

## D.1 Fuzzy-Mengen

### D.1.1 Grundidee

Das Konzept der Fuzzy-Logik wurde erstmals im Jahr 1965 durch Zadeh [Zad65] zur Modellierung linguistischer Begriffe eingeführt. Die wörtliche Übersetzung als „unscharf“läßt den Grundgedanken hinter dem Forschungsgebiet bereits erahnen. Der Ansatzpunkt der Fuzzy-Logik liegt darin einer Eingabe nicht die Zugehörigkeit zu einer Menge zu , sondern einen Grad zuzuordnen, mit dem diese Eingabe zu einer Menge gehört. Diese Vorgehensweise hat den Vorteil, dass sehr ähnliche Eingaben auch immer ähnlich kategorisiert werden können. Als Beispiel hierfür sei die Zuordnung einer Temperatur in die Kategorien warm und kalt angeführt. Bei der klassischen Betrachtung existiert ein Schwellwert, bei dem zwei aneinandergrenzende Temperaturen völlig anders zugeordnet werden. Alle Werte ab beispielsweise 21 Grad werden der Kategorie warm zugeordnet, alle darunter der Kategorie kalt.
Im Gegensatz zur klassischen Prädikatenlogik beschränkt die Fuzzy-Logik sich nicht auf die beiden Zustände wahr und falsch, sondern erweitert den Bereich auf Werte dazwischen. Hierzu wird der Zugehörigkeitswert $\mu_A(a)$ definiert, der einem Objekt einen Wert zwischen 0 und 1 zuweist. Dieser Wert gibt an zu welchem Grad eine Eigenschaft auf eine Eingabe zutrifft.

Die Zugehörigkeitsfunktion $\mu$ zu einer Menge A für eine Stützmenge X - dem Bereich der möglichen Eingaben - ist also wie folgt definiert:

| Klassische Prädikantenlogik: | Fuzzy Logik: |
|---|---|
| $\mu_A : X \rightarrow \{0, 1\}$ | $\mu_A : X \rightarrow [0, 1]$ |

### D.1.2 Linguistischer Term & linguistische Variable

In der gesprochenen Sprache werden Sachverhalte häufig unpräzise wiedergegeben. In dem folgenden Ausdruck werden drei unpräzise Informationen miteinander verknüpft.
„Wenn das Auto *kurz vor* einem Hindernis ist und die Geschwindigkeit *hoch* ist, dann *stark*bremsen“
Die linguistischen Terme *kurz vor* und *hoch* repräsentieren Werte für Messgrößen, der Term *stark* hingegen einen geeigneten Stellwert.
Eine linguistische Variable wie beispielsweise die der Geschwindigkeit wird meistens durch mehrere linguistische Terme beschrieben, deren Fuzzy-Mengen den Wertebereich der Variablen abdecken.[Gra04]

## D.2 Mitgliedschaftsfunktionen

### D.2.1 Darstellungsmöglichkeiten

Mitgliedschaftsfunktionen können auf verschiedenen Arten dargestellt werden. Falls es die Stützmenge $X = \{a_1, a_2 \dots a_n\}$ endlich ist, kann für jede Eingabe der zugehörige Wert explizit angegeben werden:

Eine weitere Möglichkeit ist die Darstellung als Vereinigung mehrerer Fuzzy-Singulärtests.

| | $x_1$ | $x_2$ | $x_3$ | ... | $x_n$ |
|---|---|---|---|---|---|
| $\mu_A(x)$ | 0,2 | 0,3 | 0,7 | | 0,2 |

**Tabelle D.1:** Mitgliedschaftstabelle

Ein Fuzzy-Singulärtest ist eine Fuzzy-Menge, deren Stützmenge aus genau einem Element besteht

A = $\mu$ / x

Hier repräsentiert A eine Menge, die einer Eingabe nur den Zugehörigkeitsgrad $\mu$ zuordnet, falls sie den Wert x entspricht. Die bereits angesprochene Vereinigung mehrerer Fuzzy-Singulärtests sieht dann folgendermaßen aus.

A = $\mu_1$ / $x_1 \dots \mu_n$ / $x_n = \sum_{i=1}^{n} \mu_n$ / $x_n$

Für den Fall einer unendlichen Stützmenge wird das Summenzeichen durch ein Integral ersetzt.

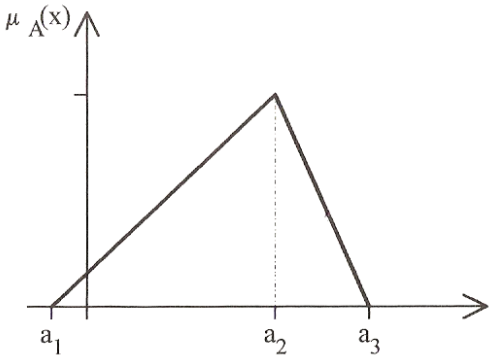
A = $\int_{x \in X} \mu_A(x)$ / $x$

Für die Bestimmung des Komplements einer Zugehörigkeitsfunktion wird in den Fuzzysingulärtests der Grad der Zugehörigkeit (meistens) durch die Differenz aus 1 und dem Zugehörigkeitsgrad der originalen Funktion ersetzt: $A^C = \int_x (1 - \mu_A(x))/x$

### D.2.2 Zugehörigkeitsfunktionen

In der Praxis haben sich verschiedene Mitgliedschaftsfunktionen bewährt. Viele dieser Funktionen weisen der Eingabe in Abhängigkeit von den Intervall, in dem sie sich befinden verschiedene Zugehörigkeitfunktionen zu. So lassen sich mit relativ einfachen Formeln verschiedene Verläufe verwirklichen.
Die erste hier eingeführt Funktion ist die so genannte Dreiecksfunktion (Abb. 45) . Sie steigt ab einem Punkt $a_1$ linear an und erreicht in einem Punkt $a_2$ ihr Maximum. Danach fällt sie linear bis sie im Punkt $a_3$ wieder den Wert 0 erreicht. Diese Zugehörigkeitsfunktion wird in der Praxis verwendet, falls für einen Bereich um einem Referenzwert $a_2$ eine Zugehörigkeit zu einer Menge erwünscht ist. Eine leichte Abwandlung der Dreiecksfunktion ist die Trapezfunktion, bei der ein komplettes Intervall statt eines einzelnen Punktes den Zugehörigkeitsgrad 1 zu einer Menge erhält.

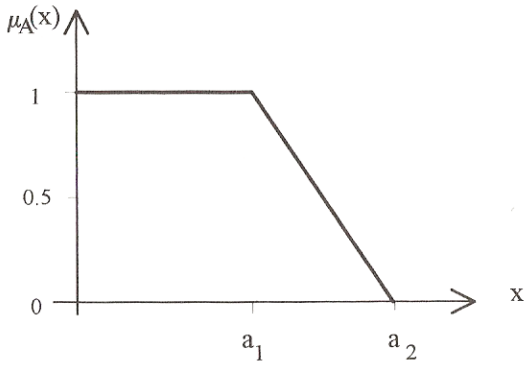
Die Γ-Funktion (Abb. 46) ordnet jedem Wert unter dem Schwellwert $a_1$ den Grad 0 zu und über dem Schwellwert $a_2$ den Wert 1 zu. In dem Intervall dazwischen steigt der




$\mu_A(x) = 0$, falls $x < a_1$
$\mu_A(x) = \frac{x-a_1}{a_2-a_1}$, falls $a_1 \leq \mathrm{x} \leq a_2$
$\mu_A(x) = \frac{a_3-x}{a_3-a_2}$, falls $a_2 \leq \mathrm{x} \leq a_3$
$\mu_A(x) = 0$, falls $x > a_3$

**Abbildung 45:** Dreiecksfunktion [Gra95]

Zugehörigkeitsgrad linear.




$\mu_A(x) = 1$, falls $x < a_1$
$\mu_A(x) = \frac{a_2-x}{a_2-a_1}$, falls $a_1 \leq \mathrm{x} \leq a_2$
$\mu_A(x) = 0$, falls $x > a_2$

**Abbildung 46:** Γ-Funktion [Gra95]

## D.3 Mengenalgebraische Funktionen

### D.3.1 Komplement

Das Komplement einer Fuzzy-Menge liefert den Grad dafür, dass ein Element nicht zu einer Menge gehört. Sie ist eine Funktion, deren Werte- und Definitionsbereich im Intervall von 0 bis 1 liegt.
$c : [0, 1] \rightarrow [0, 1]$
Eine Fuzzy-Komplementärfunktion muss sich an den Rändern genau wie deren klassisches Vorbild verhalten, so dass eine definitive Zugehörigkeit zu einer Menge einen Zugehörigkeitsgrad von 0 für das Komplement der Menge liefert. Zudem wird von der Funktion zusätzlich verlangt, dass jene monoton fallend und involutorisch ist. Häufig wird deshalb die folgende Funktion verwendet:
$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$

### D.3.2 Vereinigung & Durchschnitt

Zumal die mengenalgebraische Vereinigung und Durchschnitt nicht mehr wie bei normalen Mengen angewendet werden können, definieren wir den Begriff der t-Norm mit deren Hilfe dann Funktionen für die Mengenoperationen ermittelt werden können. Eine Abbildung von $[0,1]\times[0,1] \rightarrow [0,1]$ heißt t-Norm, falls sie folgende Eigenschaften erfüllt:

| | |
|---|---|
| Symmetrie | t(x,y) = t(y,x) |
| Assoziativität | t(x,t(y,z)) = t(t(x,y),z) |
| neutrale Elemente | t(x,1) = x |
| | t(x,0) = 0 |
| Monotonie | t(x,y) $\leq$ t(v,w), falls x $\leq$ v, y $\leq$ w |

Jede t-Norm besitzt genau eine t-Conorm, die in folgendem Zusammenhang mit jener steht:

$s_t(x,y) = 1 - t(1-x, 1-y)$
$t_s(x,y) = 1 - s(1-x, 1-y)$

Sie erfüllt dann analog zur t-Norm die Assoziativitäts-, Symmetrie- und Mononie-Eigenschaft. Bezüglich der neutralen Elemente gelten folgende Gleichungen.

| | |
|---|---|
| neutrale Elemente | s(x,0) =(0,x) = x |
| | s(x,1) = 1 |

Falls die t-Norm t, die t-Conorm s und eine Fuzzy-Negation $\neg$ dem Gesetz von De Morgan genügen, werden diese drei Operatoren zusammen auch als „duales Tripel“bezeichnet:
$\forall a, b \in [0,1] : \neg s(a,b) = t(\neg a, \neg b)$
$\forall a, b \in [0,1] : \neg t(a,b) = s(\neg a, \neg b)$

Jede t-Norm kann zur Ermittlung des Durchschnitts und jede t-Conorm für die Vereinigung zweier oder mehrerer Fuzzy-Mengen verwendet werden. Eine sehr einfache t-Norm ist die Minimum-Funktion (Abb. 47) $t_d(x,y) = min(x,y)$ und die dazugehörige t-Conorm die Maximum-Funktion (Abb. 48) $s_v(x,y) = max(x,y)$:

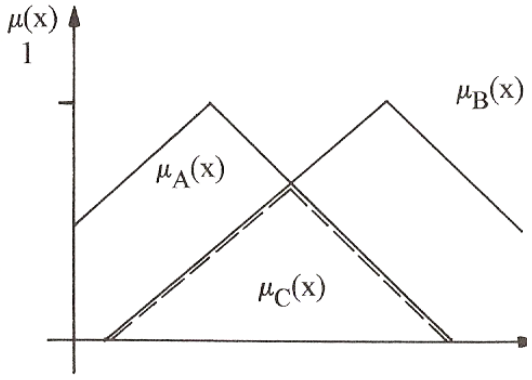



**Abbildung 47:** Minimumfunktion [Gra95]

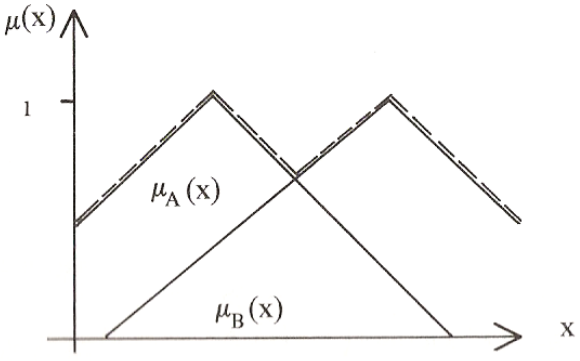



**Abbildung 48:** Maximumfunktion [Gra95]

### D.3.3 Alternative Vereinigungs- & Durchschnittfunktionen

In der Wissenschaft wurden weitere t-Normen und t-Conormen erforscht. Zu den bekanntesten und in der Praxis am häufigsten verwendeten t-Normen gehören das Hamacher, das Algebraische und das Einstein Produkt. Jene bilden mit der gleichnamigen Summe und der Standardnegation N(x)=1-x ein duales Tripel. Welche Durchschnitts- und Vereinigungsfunktion in einer praktischen Anwendung bessere Ergebnisse erzielt, lässt sich nicht pauschalisieren, so dass beim Entwurf eines Fuzzy-Systems mehrere dieser Funktionen getestet werden sollten. Jedoch besitzen diese drei Durchschnitts- und Vereinigungsfunktionen gegenüber der Minimum- und Maximum-Funktion den Vorteil der Differenzierbarkeit. Insbesondere in Neuro-Fuzzy-Sytemen, die später behandelt werden, erweist sich dies als großer Vorteil.

| | |
|---|---|
| Hamacher Produkt | $t_h(x,y) = \frac{x \cdot y}{x+y-x \cdot y}$ |
| Hamacher Summe | $s_h(x,y) = \frac{x+y-2 \cdot x \cdot y}{1-x \cdot y}$ |
| Algebraisches Produkt | $t_a(x,y) = x \cdot y$ |
| Algebraische Summe | $s_a(x,y) = x + y - x \cdot y$ |
| Einstein Produkt $t_e$ | $t_e(x,y) = \frac{x \cdot y}{1+(1-x)+(1-y)}$ |
| Einstein Summe $s_e$ | $s_e(x,y) = \frac{x+y}{1+x \cdot y}$ |

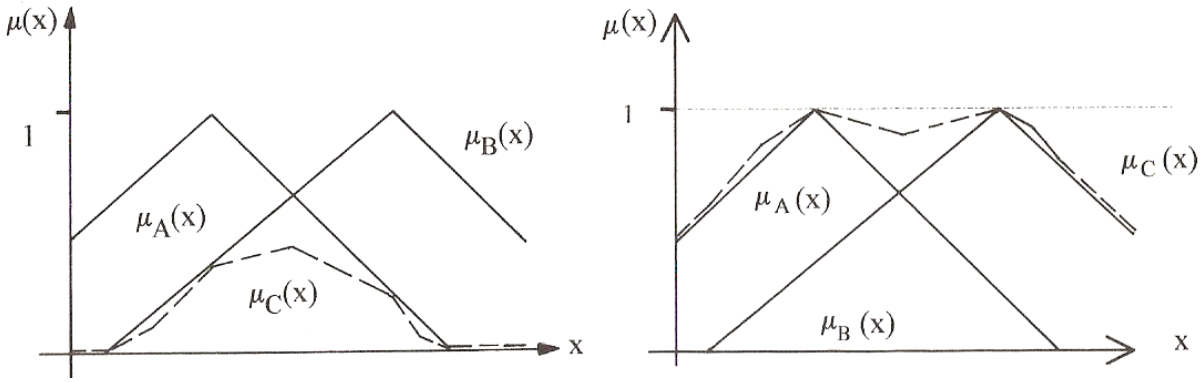



**Abbildung 49:** Algebraisches Produkt und algebraische Summe [Gra95]

Eine Fuzzy-Relation von einer Menge X in eine andere Menge Y ist eine Relation mit dem kartesischen Produkt der beiden Mengen als Grundbereich. Diese Relation auf den beiden Mengen liefert also inwiefern eine bestimmte Eigenschaft des ersten Datum auch auf das zweite Datum zutreffend ist.
Bei Verwendung der Minimumfunktion als Durchschnitt erhält man also folgende Formel:
$\mu_{M_1 \times \cdots \times M_n}(x_1, \ldots, x_n) := min\{\mu_{M_1}(x_1), \ldots, \mu_{M_n}(x_n)\}$

Die Verkettung bzw. das Relationsprodukt (kompositorische Inferenzregel) für zwei Fuzzy-Relationen mit den Stützmengen $R \subseteq X \times Y$ und $S \subseteq Y \times Z$ kann folgendermaßen ermittelt werden:

$R \circ S = sup_{y \in Y}\{min(\mu_R(x,y)\mu_A(y,z))\} \forall (x,z) \in X \times Z$

## D.4 Aufbau von Fuzzy-Systemen

Fuzzy-Systeme besitzen eine Menge von Eingangssignalen und verarbeiten diese zu einer Menge von Ausgangssignalen. Entgegen der Bezeichnung handelt es sich sowohl bei den Eingangs- und Ausgangssignalen um scharfe Signale, die für die Verarbeitung erst in unscharfe Signale umgewandelt werden. Auf diesen unscharfen Daten kann dann ein Regelsystem angewendet werden, bevor diese in scharfe Signale zurückgewandelt werden.

### D.4.1 Fuzzyfizierung

Die erste Verarbeitungseinheit wandelt scharfe Signale in unscharfe Signale um. Dieser Verarbeitungsschritt wird als Fuzzifizierung bezeichnet. Hierzu wird für exakte Daten ein Zugehörigkeitsgrad für vorgegebene Eigenschaften ermittelt. So kann beispielsweise eine Temperatur von 16 Grad zu 20% der Eigenschaft kühl und zu 80% der Eigenschaft angenehm zugeordnet werden. An dieser Stelle sei erwähnt, dass die Summe der Zugehörigkeitsgrade zu einzelnen Eigenschaften sowohl niedriger als auch höher als 100% sein kann.

### D.4.2 Inferenz

Ein wichtiges Werkzeug für Deduktionen sind der Modus Ponens und der Modus Tollens. Der Modus Ponens ordnet Eingabewerten mit Hilfe eines Regelwerks eine Konsequenz zu.

| | |
|---|---|
| $p \Rightarrow q$ | Wenn die Sonne scheint, habe ich gute Laune |
| p | Die Sonne scheint |
| q | Ich habe gute Laune |

Der Modus Tollens liefert durch eine nicht gegebene Konsequenz, dass die Vorbedingung einer Regel ebenfalls nicht erfüllt sein kann.

| | |
|---|---|
| $p \Rightarrow q$ | Wenn die Sonne scheint, habe ich gute Laune |
| ¬q | Ich habe keine gute Laune |
| ¬p | Die Sonne scheint nicht |

Der Hypothetische Syllogismus erlaubt es anhand einer Folgerung weitere Konsequenzen zu folgern.
$\{p \rightarrow q, q \rightarrow r\} \Rightarrow p \rightarrow r$

In der Fuzzy-Logik werden diese Implikationen in generalisierter Form eingeführt. Der generalisierte Modus Ponens besitzt dabei die Form IF X is A, THEN Y is B, wobei A eine Fuzzy-Menge über X ist und B eine Fuzzy-Menge über Y.

| | |
|---|---|
| A $\Rightarrow B$ | IF X is A, THEN Y is B |
| $\mu_{A'}(x)$ | X is A' |
| $\mu_{B'}(y)$ | Y is B' |

Zur Ermittlung des Ergebnisses, wird eine Fuzzy-Menge R über X $\times$ Y definiert, mit folgenden Zugehörigkeitsgraden:
$\forall (x, y) \in X \times Y : R(x, y) = Imp(\mu_A(x), \mu_B(y))$
mit $Imp : [0, 1] \times [0, 1] \rightarrow [0, 1]$

Eine “geeignete“Fuzzy-Implikation ist:
Imp(a, b) = min{ 1, 1 - a + b }

Mit Hilfe von R kann nun, das Ergebnis der Implikation ermittelt werden.
$B' = A' \circ R$

Der Modus Tollens wird auf die Fuzzy-Logik durch den generalisierten Modus Ponens übertragen.

| | |
|---|---|
| $A \Rightarrow B$ | IF X is A, THEN Y is B |
| $\mu_{B'}(y)$ | Y is B’ |
| $\mu_{A'}(x)$ | X is A’ |

Nun kann das Ergebnis der Implikation mit folgender Formel berechnet werden:
$B' \circ R^{-1} = A'$

Der generalisierte Hypothetische Sylogismus wird durch das Relationsprodukt von $R_1$ und $R_2$ aus den beiden Konklusionen ermittelt: $R_1$ und $R_2 = R_G$

Den Kern eines Fuzzy-Systemes bildet eine Menge von Regeln, die auf die fuzzifizierten Daten angewendet werden. Dieser Teil ermöglicht dem Leser einen Einblick in die Umsetzung von Wenn-Dann-Bedingungen in der Fuzzy-Logik. Es ist absolut nicht trivial, wie eine derartige Bedingung in ein Fuzzy-System integriert werden kann, da A und B anders als in der Prädikaten-Logik keine binären Wahrheitswerte sind:

WENN A DANN B

Zuerst wird diese Formel um ein Sonst-Klausel erweitert.

WENN A DANN B SONST NOT B = $(A \times B) + (\overline{A} \times \overline{B})$

Durch die angegebene Mengenalgebraische Funktion kann die Bedingung leicht umgesetzt werden.
Beispiel Zimmertemperatur: (abgeleitet aus [Gra95]) Gegeben seien folgende Zuordnungsfunktionen:
A = niedrige Raumtemperatur = 1/17 + 0,8/19 + 0,6/21 + 0,4/23 + 0,2/25
B = hohe Ventilatorauslastung = [0,2 , 0,4 , 0,6 , 0,8 , 1]
C = keine sehr hohe Ventilatorauslastung = [0,96 , 0,84 , 0,64 , 0,36 , 0]
Je nach höhe der Temperatur wird dem Raum eine Zugehörigkeit zu einer Gruppe zugeordndet. Trivialerweise ist die Zugehörigkeitsfunktion in diesem Fall monoton fallend.

Darauf soll die Inferenzregel
WENN niedrige Raumtemperatur DANN hohe Ventilatorauslastung SONST keine sehr hohe Ventilatorauslastung
angewendet werden.

Zuerst werden hierzu die Kreuzprodukte $A \times B$ und $\overline{A} \times C$ berechnet.

Zur Erinnerung: $\mu_{A \times B} = min(\mu_A(u), \mu_B(v))$

$$A \times B = \begin{pmatrix} 0,2 & 0,4 & 0,6 & 0,8 & 1,0 \\ 0,2 & 0,4 & 0,6 & 0,8 & 0,8 \\ 0,2 & 0,4 & 0,6 & 0,6 & 0,6 \\ 0,2 & 0,4 & 0,4 & 0,4 & 0,4 \\ 0,2 & 0,2 & 0,2 & 0,2 & 0,2 \end{pmatrix}$$

$$\overline{A} \times C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0,2 & 0,2 & 0,2 & 0,2 & 0 \\ 0,4 & 0,4 & 0,4 & 0,36 & 0 \\ 0,6 & 0,6 & 0,6 & 0,36 & 0 \\ 0,8 & 0,8 & 0,64 & 0,36 & 0 \end{pmatrix}$$

mit $\overline{A} = 0/1 + 0{,}2/2 + 0{,}4/3 + 0{,}6/4 + 0{,}8/5$

Die anschliessende Vereinigung führt dann zu folgendem Ergebnis:

$$A \times B \text{ und } \overline{A} \times C = \begin{pmatrix} 0,2 & 0,4 & 0,6 & 0,8 & 1 \\ 0,2 & 0,4 & 0,6 & 0,8 & 0,8 \\ 0,4 & 0,4 & 0,6 & 0,6 & 0,6 \\ 0,6 & 0,6 & 0,6 & 0,4 & 0,4 \\ 0,8 & 0,8 & 0,64 & 0,36 & 0,2 \end{pmatrix}$$

Die Komposition $y = x \circ R$ liefert dann das Ergebnis:
y = [0,36 0,4 0,6 0,8 1]

### D.4.3 Defuzzifizierung

Nachdem die unscharfen Daten verarbeitet wurden, müssen diese wieder in scharfe Signale zurückgewandelt werden um eindeutige Anweisungen zu erzeugen. Diese Transformation wird in der Wissenschaft als Defuzzifizierung oder auch Defuzifikation bezeichet. Ein verbreitetes Verfahren zur Defuzzifizierung ist die Schwerpunkt-Methode. Als Ausgangswert $y_0$ wird der Abzissenwert des Schwerpunktes der Fläche unterhalb der Zugehörigkeitsfunktion $\mu_B(y)$, $y \in Y$ berechnet [Bot95]. Die Integration über Y liefert dann folgende Lösung:

$$y_0 = \frac{\int_Y y \mu_B(y) dy}{\int_Y \mu_B(y) dy}$$

In der Praxis wird dieses Integral wegen des hohen Rechenaufwandes meistens durch eine Summe angenähert.

### D.4.4 Design von Fuzzy-Systemen

Die wohl am häufigsten angewendete Methode für das Design, entwickelt durch Mandami (Abb. 50), unterteilt das Design in vier Schritte. Im ersten Schritt wird mit der Regelbasis der Kern eines Fuzzy-Systems entworfen. Anschließend wird die Fuzzifizierung der Eingangssignale umgesetzt. Die im ersten Schritt spezifizierten Regeln werden nun durch ein Verknüpfungs- / Entscheidungsmodul zu einer Einheit zusammengefasst. Der vierte Schritt umfasst die Erarbeitung einer geeigneten Defuzzifizierungsstrategie.

Häufig findet dieses Verfahren Anwendung in der Regelungstechnik. Hier wird dann eine eventuelle Abweichung von einem Referenzwert, wieder als Eingabe in das System eingespeist.

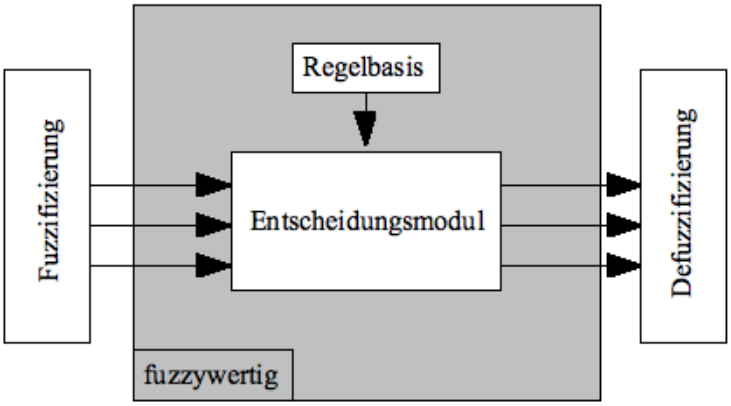



**Abbildung 50:** Design eines Fuzzy-Systems (Mandami)

## D.5 Analyse von Fuzzy-Systemen

Nachdem nun auf alle Abschnitte von Fuzzysystemen eingegangen wurde und das prinzipielle Vorgehen von Fuzzy-Systemen deutlich geworden ist, wird mit eine Analyse der Vor- und Nachteile von Fuzzy-Systemen, sowie Möglichkeiten der Weiterentwicklung fortgesetzt.

### D.5.1 Vorteile

Ein wesentlicher Vorteil von Fuzzy-Systemen liegt darin, dass kein mathematisches Modell für die Verwirklichung eines derartigen Systems notwendig ist. Der Verzicht auf ein mathematisches Modell kann den Aufwand für die Fuzzy-System-Erstellung drastisch verringern. In vielen Anwendungsfällen werden sehr komplexe Systeme betrachtet, deren Verwirklichung als mathematisches Modell aufgrund der Komplexität häufig sogar gar unmöglich ist. Ein weiterer Vorteil liegt darin, dass (Regel-)Wissen aus der Praxis direkt in die Entwicklung einfließen kann. Anhand des Regelsystems können mit beschränktem Aufwand neue Systeme entwickelt werden. [BKKN95]

### D.5.2 Nachteile

Die Anwendung dieses Verfahrens kann nur in Bereichen durchgeführt werden, in der Wissen über die Regeln des Systems vorhanden sind. Ein Fuzzy System kann also nicht allein anhand von Eingabedaten und Ausgabedaten erstellt werden ohne die internen Mechanismen zu kennen. Dieser sehr statische Ansatz führt dazu, dass (bisher) keine formalen Methoden für die Steigerung der Qualität des Systems entwickelt werden konnten. Das einmal vorgegebene Regelsystem hindert das System daran aus vergangenen Verläufen zusätzliche Informationen zu erlangen.[BKKN95]

### D.5.3 Weiterentwicklung

Nach der Analyse der Vor- und Nachteile gilt es für Weiterentwicklungen die positiven Aspekte zu erhalten oder sogar zu fördern, während die negativen Eigenschaften des Systems beseitigt werden. Daher erscheint es sinnvoll Methoden anderer Verfahren der

Computational Intelligence in das System zu integrieren. Ein weiteres sehr aussichtsreiches Forschungsgebiet der Computational Intelligence sind die neuronalen Netze. Ein nahliegender Ansatz liegt nun darin die positiven Aspekte des einen Verfahrens in das andere zu integrieren oder beide Verfahren zu verschmelzen. Die Kombination zu einem Neuro-Fuzzy-System verspricht aus Sicht des Fuzzy-Systems primär die Integration der Lernfähigkeit.

## D.6 Neuro-Fuzzy-System

Das Design eines Neuro-Fuzzy-Systems basiert üblicherweise auf der Repräsentation des Fuzzy-Systems durch die Architektur des neuronalen Netzes. Allerdings basieren Neuronale Netze in der Regel auf Gradientenabstiegsverfahren und erfordern daher differenzierbare Funktionen. Bei der ersten der beiden Lösungsmöglichkeiten, werden alle nicht-differenzierbaren Funktionen durch Funktionen ersetzt, die dieser Anforderung genügen. Die Alternative ist die Verwendung eines Lernverfahrens, das nicht gradientenbasiert ist.

Ein Neuro-Fuzzy-Netz kann in der Regel in verschiedene Schichten unterteilt werden (Abb. 51) . Die Eingangsschicht erhält die Eingangsvariablen und verteilt die zugeordneten Zugehörigkeitsgrade an die nächste - als Antezedenzschicht bezeichnete - Schicht. In dieser Schicht werden diese Grade für die Regelschicht angepasst. Diese Anpassung erfolgt im einfachen 3-Schichten-Modell durch die Multiplikation mit einer Konstante (Antezedenzgewicht). Nachdem die Regelschicht durchlaufen wurde, werden die Signale in einer weiteren Schicht (Konsequenzschicht) bzw. durch Multiplikation mit einer Konstante (Konsequenzgewicht) für die Ausgabeschicht manipuliert. Die Ausgabeschicht defuzzifiziert die Signale dann in scharfe Ausgangssignale. Bei der Optimierung durch einen Lernalgorithmus werden nur die Parameter der Antezendenz- und Konsequenzschicht bzw. im 3-Schichten-Modell die Antezendenz- und Konsequenzgewichte verändert.

## D.7 Verwendung von Fuzzy-Systemen in Spielen

Fuzzy-Systeme sind genau wie evolutionäre Algorithmen und neuronale Netze für Videospiele geeignet. Durch die Reduzierung eines komplexen mathematischen Modells auf eine Annäherung, die sehr schnell ausgerechnet werden kann, haben sie sich insbesondere bei Echtzeitspielen bewehrt. Da Echtzeitspiele häufig eine harte obere Schranke für Kalkulationen haben, kann in der Regel nicht das komplette System betrachtet werden. Minimale Abweichungen von einem optimalen Verhalten sind für den Anwender meistens nicht erkennbar und ersparen hohe Aufwände.

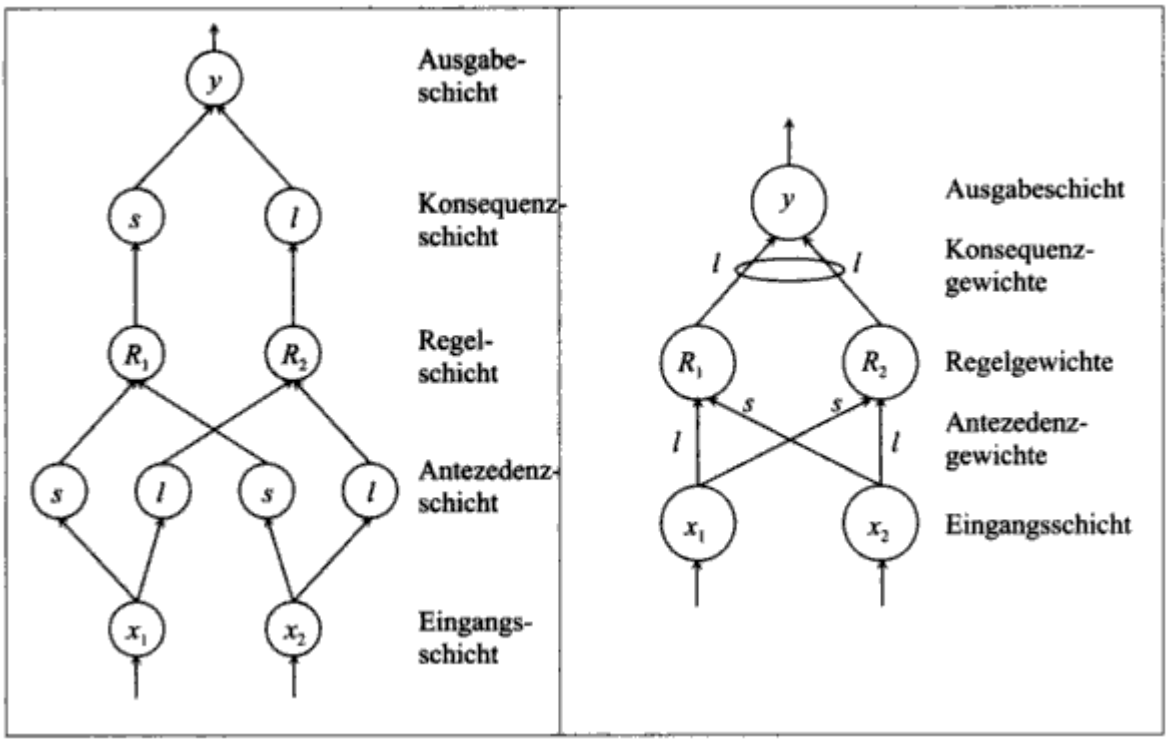



**Abbildung 51:** Aufbau eines Neuro-Fuzzy-Systems [BKKN95]

# E Künstliche Intelligenz in den Spielen Poker, Diplomacy und Junta

## E.1 Einführung

In meiner Ausarbeitung zum Thema Künstliche Intelligenz in den Spielen Poker, Diplomacy und Junta werde ich zunächst auf die Unterschiede zwischen Schach und 4 Gewinnt auf der einen Seite sowie Diplomacy Poker und Junta auf der anderen Seite eingehen. Danach werde ich die Spiele Poker und Diplomacy im Einzelnen etwas genauer betrachten und jeweils einfache und erweiterte Strategien vorstellen. Anschliessend werde ich die Spiele gegenüberstellen und probieren Gemeinsamkeiten sowie Unterschiede herauszuarbeiten. Abschliessend werde ich die wichtigsten Informationen in Form eines Fazits noch einmal zusammenfassen. Um mich in dieser Arbeit auf die KI-Aspekte konzentrieren zu können und da die Spiele bereits im Vorfeld ausgetestet werden sollten, setze ich die Kenntnisse der Spielregeln bei Texas Hold'em, Diplomacy und Junta als bekannt voraus.

## E.2 Vergleich der Spiele Schach, 4 Gewinnt mit den Spielen Poker etc.

Seit Jahren schon gibt es sehr starke Computergegner für Spiele wie Schach oder 4 Gewinnt, jedoch ist es für die Spiele, die wir betrachten möchten, weitaus schwieriger geeignete Computergegner zu entwerfen. Zwar ist man durch immer schneller werdende Rechner und durch Einsetzen von besseren Verfahren in der Lage immer grössere Suchräume verarbeiten zu können, jedoch stoesst man unweigerlich auf einige Schwierigkeiten, die man im wesentlichen auf folgende Punkte zurückführen kann:

- Der Mitspieler kennt nicht die gesamte Spielsituation (vollständige Information), sondern nur einen Teil der Situation (partielle Information). So kennt man beim Poker nur die eigene Hand und die der anderen Spieler nicht.
- Insbesondere bei Poker handelt es sich um ein Glücksspiel, bei dem es auch sehr auf die zufällig verteilten Karten ankommt. Der Verlauf des Spiels ist nicht etwa deterministisch bestimmt sondern wird mitbestimmt durch ein Zufallsereignis.
- Bei den Spielen Junta und Diplomacy kommt zudem noch hinzu, dass die Kommunikation mit den Mitspielern eine sehr grosse Rolle spielt. Die Analyse und die Interaktion mit den anderen Mitspielern ist für den Computer sehr schwierig.

Möchte man einen guten Computergegner für unsere Spiele entwerfen, wird man nicht an der Behandlung dieser Schwierigkeiten vorbeikommen.

## E.3 Künstliche Intelligenz im Spiel Poker

### E.3.1 Der perfekte Pokerspieler

Naive Ansätze beim Entwurf für Computergegner für Poker bestanden darin, einfache Charaktereigenschaften eines Mitspielers darzustellen. Hierbei versucht man Charaktereigenschaften wie z.B. vorsichtig oder agressiv zu simulieren. Ein vorsichtiger Spieler ist also eher bereit ein Blatt aufzugeben als ein agressiver Spieler. Ein aktuell starker

Computergegner enthält verschiedene Komponenten, um das Spielgeschehen analysieren zu können und gut agieren zu können. Zu diesen Komponenten zählen:

- Handstärke: Der Bot muss in der Lage sein die Stärke seine Hand einschätzen zu können. Nach jeder neuen Karte muss die Handstärke neu berechnet werden.
- Handpotential: Der Bot kann das Potential seiner Hand bestimmen. Vier ♡ - Karten haben zwar keine große Handstärke, jedoch durch die Chance auf einen Flush ein hohes Handpotential.
- Setzstrategie: Die Setzstrategie bestimmt, ob man in einer bestimmten Situation folden, callen, checken oder raisen soll. Hierbei werden die Gewinnchancen dem möglichen Gewinn gegenübergestellt. Es ist eine wichtige Entscheidung ob etwa ein hoher möglicher Gewinn ein großes Risiko wert ist.
- Bluffen: Durch geschicktes Bluffen sollte der Bot in der Lage sein auch aus einer schwachen Hand Profit zu schlagen.
- Gegnermodellierung: Wenn man in der Lage ist den Gegner gut einzuschätzen kann dies ein wertvoller Vorteil sein.
- Unvorhersehbarkeit: Wichtig ist, dass die eigene Vorgehensweise nicht leicht durchschaubar für die Gegner ist, da diese sonst leicht eine Gegenstrategie entwickeln können.

### E.3.2 Ein mathematischer Ansatz

Das Verfahren besteht aus zwei Komponenten. Einer zur Bewertung der Hand und die andere für die Setzstrategie. Die Setstrategie wird sowohl durch die Einschätzung des Gegners als auch durch das Verhältnis zwischen dem zu bringenden Einsatz und des möglichen Gewinns bestimmt. zur Bestimmung des Wertes der aktuellen Hand benötigt man

- Pre-flop Evaluation: Für die beiden Startkarten gibt es $\binom{52}{2} = 1326$ mögliche Kombinationen, aber nur 169 unterschiedliche Handtypen. (Hierzu ist anzumerken, dass beim Texas Hold'em Poker der Wert einer Karte nicht davon abhängt welche Farbe sie hat. Starthände wie ♣ - Ass und ♡ - Ass sowie ♠ - Ass und ♢ - Ass sind also gleichwertig.) Für jede dieser 169 möglichen Hände wurde im Vorfeld auf Basis der Kombinationsmöglichkeiten zur Straße, Flush etc. die statistische Gewinnchance berechnet. Ein Paar Asse als Starthand hat statistisch die höchste Gewinnchance und die Kombination Zwei und Sieben (unsuited) die geringste.
- Handbewertung: Zu einer gegebenen Hand werden die Anzahl der möglichen Hände gezählt, die besser sind als die eigene, sowie die Anzahl der Hände die schwächer sind. Nun kann man ein prozentuales Ranking errechnen, welches die potentielle Handstärke widerspiegelt. Für die Starthand ♢ - Ass und ♣ - Dame kann man so eine Handstärke von $0,585$ errechnen. Diese Berechnung gilt allerdings nur bei einem Gegenspieler. Sollten sich mehrere Mitspieler im Spiel befinden verändert sich die Handstärke. Bei fünf Gegenspielern mit zufälligen Händen wäre unsere Handstärke nurnoch 0.069.
- Handpotential: Um das Potential einer Hand zu bestimmen, also die Chan- ce mit noch kommenden Karten die beste Hand am Tisch zu bekommen, wird ähnlich wie bei der Handbewertung vorgegangen. Es erhoht sich lediglich der Rechenaufwand, da man fur alle moglichen nächsten aufzudeckenden Karten (Turn, River) eine Berechnung vornehmen muss, ob man mit diesen Karte eine bessere, schlechtere oder gleichwertige Hand im Vergleich zu der des Gegenspielers bekommt. Betrachten wir beispielhaft die folgende Spielsituation: Wir haben ♢ - Ass, ♣ - Dame auf der

Hand sowie ♡ - Drei, ♣ - Vier und ♡ - Bube im Flop. Falls wir bis jetzt nicht die beste Hand hatten ergeben sich 91981 Kombinationen von Karten, die uns die beste Hand geben würden. Allerdings ergeben sich auf der anderen Seite 1036 Kombinationen die eine gleichwertige Hand ergeben und 345543 Kombinationen die uns eine schlechtere Hand geben als unserem Gegner. Die Chance, dass unsere Hand die Gewinnerhand ist liegt also bei nur ungefähr $0,21$. Dieser Prozentwert ist die Maßzahl für unser Handpotential.

- Gewichtung der Wahrscheinlichkeiten: Bis jetzt sind wir davon ausgegangen, dass alle gegnerischen Hände gleich wahrscheinlich sind. In der Realität handelt es sich leider um eine weniger homogene Verteilung. Oft ist es so, dass ein Spieler mit einer schwachen Hand wie z.B. ♢ - Vier und ♣ - Bube schon vor dem Flop foldet. Auf der anderen Seite ist gerade diese Starthand in Kombination mit unserem Beispielflop ♡ - Drei, ♣ - Vier und ♡ - Bube eine sehr starke Hand, welches die Handbewertung verzerrt. Die Genauigkeit unserer Annahmen hängt stark von der Analyse des Gegners ab. Es werden Gewichte eingeführt für jede mögliche Starthand des Gegners. Diese Gewichte werden dann je nach Spielverhalten des Gegners angepasst. Zum Beispiel macht ein Spieler der beim Flop erhöht den Eindruck, dass er eine starke Hand hat, was sich in den Gewichten widerspiegeln sollte.

- Setzstrategie: Für unsere Setzstrategie kombinieren wir die für die Stärke der Hand bei $n$ Gegenspielern ($HS(n)$) und das Handpotential ($HP(n)$) errechneten Werte. Die so errechnete effektive Handstärke ($EHS(n)$) ist ein Maß für die effektive Stärke unserer Hand.

  Für unsere Setzstrategie kombinieren wir die fur die Staerke der Hand bei $n$ Gegenspielern ($HS(n)$) und das Handpotential ($HP(n)$) errechneten Werte. Die so errechnete effektive Handstarke ($EHS(n)$) ist ein Maß für die effektive Starke unserer Hand.

  $$EHS(n) = Hs(n) + (1 - HS(n)) * HP(n))$$

  Die Formel drückt aus, dass wir entweder vorne liegen - mit einer Wahrscheinlichkeit von $HS(n)$ - und wenn nicht ein gewisses Potential ($HP(n)$) haben nach dem Aufdecken der Karten noch die beste Hand zu bekommen. Rein statistisch - ohne Bluffen - gewinnt man mit einer $EHS(n) > 0,5$ mehr als die Hälfte der Spiele. Deshalb macht es Sinn erst ab dieser EHS zu raisen. Beim Checken im Gegensatz zum Raisen wird anders vorgegangen. Um zu bestimmen, ob man in gegebener Runde checken soll oder nicht, berechnen wir zunächst das Verhältnis von Einatz zum gesamten Inhalt des Pots. Der Begriff in der Pokerwelt hierfür ist Pot Odds(engl. fuer Topf-Wettchancen).

  $$PotOdds = unserEinsatz/(Topfinhalt + unserEinsatz)$$

  Wir callen wenn das Handpotential größer ist als $PotOdds$.

### E.3.3 Das Problem der Teilinformation

Wie bereits am Anfang angeschnitten, haben wir beim Poker im Gegensatz zu klassischen Spielen wie Schach nur einen begrenzten Teil der gesamten Spielinformationen zu Verfügung. Das Spiel Poker kann durch einen unvollständigen Informationsspielbaum (imperfect information game tree) dargestellt werden, welcher in Informationssets gruppierte Wahrscheinlichkeits- und Entscheidungs-Knoten hat. Da die Knoten des Baumes allerdings nicht unabhängig sind, ist die Evaluierung des Spielbaums kompliziert.

Man ist deshalb bemüht, vereinfachte Formen des Spiels zu untersuchen, von denen man sich dennoch erhofft verallgemeinerte Erkenntisse für das gesamte Problem abstrahieren

zu können. Ziel ist es eine brauchbare Approximation der optimalen Strategie für das nicht vereinfachte Spiel Poker zu finden. Durch Abstraktion ist es bereits gelungen den Suchraum von $O(10^{18})$ auf $O(10^7)$ zu reduzieren ohne die wichtigsten Aspekte im Spiel zu verlieren. Zu den wesentlichen Vereinfachungen zählt es die vier bets pro Runde auf drei zu reduzieren sowie Spielsituationen in Äquivalenzklassen aufzuteilen, um Berechnungen einzusparen.

## E.4 Künstliche Intelligenz im Spiel Diplomacy

### E.4.1 Über das Spiel

Das Spiel Diplomacy ist deterministisch, da das Ergebnis eines Zuges nicht vom Zufall wie etwa dem Würfeln eines Würfels abhängig ist. Beim Spielen des Spiels muss man aber mit partiellen Informationen auskommen, da die gleichzeitig von den Mitspielern ausgeführten Züge unbekannt sind und man auch nicht die Absprachen kennt, die die Mitspieler untereinander getroffen haben. Das heißt, dass obwohl das Spiel deterministisch ist, man nicht in der Lage die Auswirkung der Spielzüge der Spieler auch nur einen Zug im vorraus zu bestimmen. Wie bei unseren beiden anderen Spielen Junta und Poker ist es der Mangel an Informationen der es für den Computergegner so schwierig macht gute Entscheidungen zu treffen. Offensichtlich spielt bei Diplomacy die Diplomatie, das Analysieren und Verhandeln mit den anderen Mitspielern eine sehr große Rolle. Der Spieler muss in der Lage sein die Absichten der Mitspieler zu erkennen und möglichst geschickt versuchen die Mitspieler unbemerkt gegeneinander auszuspielen. Da es sich bei Diplomacy um ein Null-Summen-Spiel handelt, es also nicht möglich ist selber stärker zu werden ohne jemand anderen dabei zu schwächen, kann man - wenn man gewinnen möchte - Konflikte mit den Mitspielern nicht vermeiden. Menschliche Mitspieler haben bei Diplomacy den Vorteil, dass sie miteinander Verhandeln können, Allianzen bilden und Konflikte beseitigen können. All diese Punkte stellen für einen Computergegner eine große Herausforderung da.

### E.4.2 DAIDE

DAIDE (Diplomacy AI Development Environment) ist eine von der DipAI genannten Organisation entwickelte Umgebung, welche die Erstellung von Computerspielern für das Spiel erleichtern soll. Es stellt ein Interface und Tools zur Verfügung, welche die Kommunikation über ein Netzwerk und die Anwendung von Verhandlungen im Spiel vereinfacht . Es ermöglicht es dem Anwender ausserdem laufende Spiele von Computergegnern über ein Mapper-Tool zu beobachten. Da viele andere Bots für Diplomacy diese Plattform nutzen, ist es möglich die eigene KI gegen die anderen antreten zu lassen und so zu testen. Die DAIDE Syntax definiert ein Kommunikationsprotokoll für die Verhandlungen zwischen den Spielern. Hierfür wurden die Arten der Kommunikation, welche zwischen den Spielern stattfinden analysiert und es ermöglicht dem Spieler mit den Computerspielern zu interagieren, ohne dass die Computerspieler natürliche Sprache verstehen müssen. Das diplomatische Geschick bzw. die Interaktionsstärke der Computergegner wird durch eine Zahl repräsentiert, welche den Umfang der Verhandlungsmöglichkeiten des Gegners widerspiegelt. Die diplomatische Stärke startet bei einem Wert von 0 und kann Werte bis 130 annehmen. 0 bedeutet hierbei keine Möglichkeit zur Verhandlung (diese Ausdrucksstärke ist bei vielen Ansätzen populär, die sich nur auf die Auswertung der aktuellen Spielsituation mit Hilfe von Mathematik und Statistik konzentrieren) und ein Wert von 130, bei welchem der Computergegner in der Lage ist selbst komplexeste Intrigen zu spinnen, bedingte Verträge einzugehen und sehr menschenähnlich zu handeln. Der Wert von 8000 - als ultimatives Ziel der Forschung

auf dem Gebiet - nimmt eine Sonderposition ein. Er steht für eine gegenseitige Kommunikation auf Basis von frei geschriebenem Text, der keine bestimmte Form haben muss.

### E.4.3 Existierende KIs

Für das Spiel Diplomacy wurden mit Hilfe der DAIDE Plattform schon diverse Computergegner unterschiedlicher Güte erschaffen. Viele von ihnen verzichten ganz auf Diplomatie ( haben eine diplomatische Stärke von 0), andere hingegen legen mehr Wert auf Diplomatie. Dennoch wird bei allen aktuellen Ansätzen eine diplomatische Stärke von 20 nicht überschritten. Einige Ansätze sind aber schon in der Lage Allianzen zu bilden und geschickte Abwägungen zu treffen, welches diplomatische Verhalten zum gegebenen Zeitpunkt sinnvoll ist. Sie sind außerdem in der Lage Allianzen zu einem geeigneten Zeitpuntkt zu brechen, was neue taktische Möglichkeiten eröffnet. Im folgenden werde ich drei Bots vorstellen, welche ich repräsentativ für die Menge der Möglichkeiten für Diplomacy-Bots halte.

#### DumbBot

DumBot ist ein sehr populärer strategischer (diplomatische Stärke 0) Bot, der in der Lage ist schwache Gegenspieler zu besiegen. Er ist relativ einfach strukturiert, nutzt keine Diplomatie, sondern setzt auf reine Strategie und heuristische Methoden. Der Bot weist jedem Knoten auf der Karte am Anfang des Zuges einen bestimmten Wert zu. Dieser Wert hängt von Parametern ab wie z.B. wem der Knoten gehört, wie stark dieser Spieler ist, wie stark oder schwach er bewacht wird und wie groß die Chancen sind mit seiner Armee den Knoten erfolgreich zu erreichen. Es gibt außerdem eine Wechselwirkung zwischen adjazenten Knoten, wodurch probiert wird dem Bot eine bessere Übersicht über das gesamte Spielgeschehen zu geben statt nur einer lokalen Betrachtung der Situation. Das Zugverhalten sieht nun vor, dass die am besten geeignete Einheit ausgesucht wird und diese zu dem Knoten mit der höchsten Wertigkeit gezogen wird. Die Stärke von DumbBot liegt im Aufbau seiner strategischen Entscheidungsfindung. Die effektive Verteilung der Armeen und Länder erlaubt einfaches und vor allem plausibles Verhalten. Der große Nachteil an diesem Bot ist natürlich, dass er, da er nicht verhandelt, niemals einen vollwertigen Menschenspieler ersetzen kann.

#### Diplomat

Der Diplomat(Bot) ist wie der Name schon erahnen lässt der Versuch, den Schwerpunkt beim Spiel von Diplomacy auf Verhandlungen zu setzen und durch List zu gewinnen. Seine Vorgehensweise kann wie folgt beschrieben werden:

- Am Anfang führt Diplomat seine Einheiten zu kleinen Gruppen zusammen und entscheidet, welche anliegenden Länder er erobern möchte.
- Danach entwirft er für jede Gruppe eine eigene Gruppenstrategie, um aus diesen Gruppenstrategien eine möglichst gute Gesamtstrategie zu entwickeln. Hierbei muss der Diplomat einen Kompromiss eingehen. Auf der einen Seite möchte er natürlich die beste Gesamtstrategie entwickeln, aber auf der anderen Seite möglichst wenige Strategien auf Anwendbarkeit untersuchen, um möglichst performant zu bleiben. Aus diesem Grund setzt der Diplomat statistische Methoden ein, um möglichst schnell fertig zu werden, ohne dabei gute Lösungen zu übersehen.
- Sobald der Diplomat mit seiner Armee ein Land erobern möchte, welches aber für ihn alleine zu stark verteidigt wird, fängt er an Diplomatie einzusetzen. Er

geht auf die anderen Mitspieler zu und bittet sie um den Gefallen ihn in diesem Kampf zu unterstützen. Als Gegenleistung bietet er dem Unterstützenden bei der Absprache an, ihm später bei einem Angriff beiseite zu stehen.

- Der Diplomat ist außerdem in der Lage einzuschätzen, wann es für ihn Sinn macht sich an Absprachen zu halten und wann er sie lieber bricht. Es kann also durchaus passieren, dass einem Spieler zwar Unterstützung zugesagt wurde für eine bestimmte Situation, er diese aber dann nicht bekommt. Sollte sich ein Spieler nicht an die Abmachung mit dem Bot halten, wird auch dies sich auf das Verhältnis zwischen Spieler und Bot auswirken. Um die Mitspieler einschätzen zu können speichert der Bot nämlich für jeden Spieler Werte für Beständigkeit und Ehrlichkeit, die sich je nach Verhalten des Spielers verändern.

Man kann den DiplomatBot als das Gegenteil von DumbBot betrachten. Seine Stärke liegt eindeutig in der Diplomatie, jedoch hat er Schwächen in der strategischen Umsetzung, da die Analyse der Mitspieler oft sehr schwierig ist.

#### The Israeli Diplomat

Der Israeli Diplomat(Bot) benutzt einen sehr komplexen, eleganten Ansatz für die Verhandlungen mit den Mitspielern. Die Struktur ist der einer Regierung im Kriegsfall, wie sie real existieren könnte, nachempfunden. Die Gegner unterhalten sich zwar nur mit einer Art Premierminister, dieser hat allerdings einen kompletten Beraterstab zur Hilfe. In dem Beraterstab sitzen Experten für die jeweiligen Fronten und für die einzelnen Gegner. Es ist also möglich die Entscheidung für die Gesamtsituation aufzuteilen, aufgeteilt zu bearbeiten und danach wieder zu einer Gesamtlösung zusammenzuführen. Dieser Ansatz gehört zur Zeit mit zu den besten und erfolgsversprechenden diplomatischen Ansätzen.

### E.4.4 Probleme beim Entwurf von KIs für Diplomatie

- großer Suchbaum : Die Anzahl der möglichen Züge in Diplomacy ist riesig. Aus diesem Grund ist die Generierung eines Suchbaums für zukünftige Züge heikel. Man kennt die Züge des Gegners nicht und da alle Züge gleichzeitig ausgeführt werden ist es noch nicht einmal möglich den Status des Spiels eine Runde in die Zukunft vorauszusagen oder gar für mehrere Runden. Die Aufgabe des Bots muss es also sein sich nicht durch die schwierige kurzfristige Planung verunsichern zu lassen, sondern eine Strategie zu entwickeln wie man auf längere Sicht erfolgreich wird.
- Schwierigkeit bei der Bestimmung des Spielstandes : Während man in deterministischen Spielen, in welchen man die komplette Spielsituation kennt ( wie in etwa Schach) sehr leicht einschätzen kann, ob man sich in einer starken oder schwachen Situation befindet, ist dies bei Diplomacy weitaus schwieriger. Es ist nämlich wichtig neben der Stärke auf dem Spielfeld, auch die diplomatischen Beziehungen zu beachten. Zwar kann es durchaus sein, dass man mehr Ländereien hat als jeder andere Spieler aber dennoch einen schweren Stand hat, da sich die schwächeren Mitspieler verbündet haben. Für den Entwurf eines guten Bot ist es also wichtig eine Methode zu entwickeln, um diplomatische Beziehungen bewerten zu können.
- Komplexität der Verhandlungen: Menschliche Spieler sind des Bots in puncto Kommunikation sehr weit überlegen. Menschliche Spieler wissen oft besser, wie man am geschicktesten verhandelt und andere Spieler austricksen kann als es Bots tun. Außerdem wissen sie, warum es sinnvoll sein kann sich Sympathien bei Mitspielern zu erarbeiten und in Situationen hilft, wo es eigentlich nicht notwendig

war. Das Vertrauen von Mensch zu Mensch ist leichter herzustellen als das Vertrauen zum den Computerspieler.

## E.5 Kuenstliche Intelligenz im Spiel Junta

Meine Suche nach Ansätzen für einen Bot für das Spiel Junta hat ergeben, dass es im Gegensatz zu dem oft behandelten Spiel Diplomacy, noch keine Publikationen zu diesem Thema gibt. Die Gründe hierfür könnten mitunter folgende sein:

- Junta ist weit weniger populär als etwa Schach,Poker oder selbst Diplomacy.
- Man schätzt den wissenschaftlichen Mehrwert des Spiels für zu gering ein oder hält das Spiel für nicht seriös genug.
- Viele beschäftigen sich lieber mit Spielen, die man mit Hilfe von deterministischen bzw. heuristischen Verfahren in den Griff bekommen kann. In Junta steht allerdings genau wie bei Diplomacy das Verhandeln mit den Mitspielern im Vordergrund, wofür es noch keine geeigneten Pauschal-Lösungen gibt.

## E.6 Vergleich der 3 Spiele

In diesem Abschnitt möchte ich mich damit beschäftigen die Gemeinsamkeiten bzw. Unterschiede der Spiele herauszuarbeiten.

- Alle drei Spiele müssen mit partiellen Informationen auskommen, da entweder die Karten der Mitspieler unbekannt sind oder es geheime und unbekannte Absprachen gibt.
- Diplomacy ist deterministisch, da das Spiel nicht von Zufallsentscheidungen abhängig ist. Poker und Junta hingegen sind es nicht, da der Spielverlauf von den gezogenen Karten und bei Junta zudem vom Würfelglück abhängt.
- Vor allem bei Diplomacy und Junta ist es nur schwer möglich den Spielstand einzuschätzen, da er nicht nur von Ressourcen (Ländern/Geld) sodern auch vom Verhältnis zu den Mitspielern abhängt.
- Da es fast unmöglich ist die Spielsituation auch nur eine Spielrunde im Voraus einzuschätzen, ist es nötig nicht kurzfristig zu planen, sondern eine mittel- bis langfristige Planung aufzustellen.
- Jedes Spiel besteht aus zwei Komponenten. Eine ist zuständig für die Strategie und die andere für die Analyse und Interaktion mit den Mitspielern.
- Wenn man bei einem Spiel auf die Realisierung der sozialen Komponente verzichtet, riskiert man ein aneinander vorbeispielen.

## E.7 Fazit

In meiner Ausarbeitung habe ich probiert die Schwierigkeiten aufzuzeigen, wel- che sich bei der Entwicklung einer KI zu den Spielen Poker, Diplomacy und Junta ergeben werden. In den Spielen ist es die Unwissenheituber die gesamte Spielsituation, die uns die Evaluierung und Analyse unserer Gegner und somit den Entwurf einer Gewinnstrategie erheblich verschwert. Am Beispiel Poker ha- be ich probiert aufzuzeigen, dass es selbst mit besten statistischen Kenntnisse nicht moglich ist eine zuverlassige Vorausageuber

das Spielgeschehen zu treffen. Je mehr Spieler mitspielen, desto hoher die Wahrscheinlichkeit, dass zumindest eine gegnerische Hand besser ist als die eigene. Statistische Gewinnchancen von unter 20 oder sogar unter 10 Prozent sind beim Poker mit mehreren Mitspielern keine Seltenheit. Es gewinnt nicht der beste Mathematiker, sondern der, der seine Gegner am besten tauschen kann. Bei Diplomacy stellt sich eindeutig das Problem der fehlenden Kommunikation. Hier ist es zwar auch der Fall , dass der Computer in der Lage ist Statistiken sehr gut auszuwerten, jedoch gegen verbundete Gegenspieler nur sehr geringe Chance hat. Die diplomatischen Fahigkeiten reichen noch nicht aus um denen eines Menschen ahnlich zu werden. Die Grenzen der rein mathematischen Berechnungen werden bei unseren Spie- len besonders deutlich. Da wir nur einen Bruchteil der gesamten Spielsitua- tion kennen, kann man sich nicht nur auf statistische Berechnungen bei der Entscheidungsfindung stutzen, sondern muss sehr an seinen Moglichkeiten zur (nonverbalen-)Kommunikation arbeiten.

# F Ludologie

Diese Ausarbeitung beschäftigt sich mit der Ludologie, einem wissenschaftlichen Forschungszweig, der sich mit kulturellen, strukturellen, kommunikativen und technischen Aspekten von Spielen, insbesondere digitalen Videospielen, auseinandersetzt.

## F.1 Einleitung

Seit den Anfängen zivilisierten Denkens liegt es in der Natur des Menschen spielen zu wollen. Heutzutage ist das Spiel eine grundlegende menschliche Aktivität, die Kreativität und im Wettkampf Energie und Kraft freisetzt. Der Kulturanthropologe Johan Huizinga definiert das Spiel in seinem Hauptwerk homo ludens [Hui94]:

> „Spiel ist eine freiwillige Handlung oder Beschäftigung, die innerhalb gewisser festgesetzter Grenzen von Zeit und Raum nach freiwillig angenommenen, aber unbedingt bindenden Regeln verrichtet wird, ihr Ziel in sich selber hat und begleitet wird von einem Gefühl der Spannung und Freud und einem Bewusstsein des ‚Andersseins' als das ‚gewöhnliche Leben'. [..] Das Spiel bringt in die unvollkommene Welt und in das verworrene Leben eine zeitweilige begrenzte Vollkommenheit. "

In dieser Ausarbeitung werden wir uns mit verschiedenen Aspekten von analogen und digitalen Spielen auseinandersetzen.

### F.1.1 Ludologie vs. Narratologie

Ludologie (lat. ludus = das Spiel) ist die Lehre vom Spiel und bezeichnet den Forschungszweig, der sich mit ästhetischen, kulturellen, kommunikativen, technischen und strukturellen Aspekten von Spielen befasst. Den Schwerpunkt machen dabei Geschichte, Entwicklung, Analyse und Theorie digitaler Spiele aus. Spieleforscher unterteilen sich in zwei Gruppen: Die erste Gruppe, die Ludologen, sind der Auffassung, dass das Spiel eine Simulation ist, in der das Agieren im Vordergrund steht. Nach ihrer Meinung zeichnet sich jedes Spiel durch die Elemente Regeln, Spielwelt sowie Gameplay aus. Auf den abstrakten Begriff des Gameplays werden wird noch in F.4.4 genauer eingehen.

Die zweite Gruppe, die Narratologen, sehen Spiele als eine weiterführende Form von Texten an, die als kommunikatives Generalkonzept verstanden werden.

### F.1.2 Einteilung der Spiele

F.G. Jünger [Jue53] unterteilt die Spiele nach ihrem Entstehungsgrund in drei Kategorien ein:

**Glücksspiele** das Spiel basiert auf dem Zufall

**Geschicklichkeitsspiele** das Spiel erfordert manuelles oder geistiges Geschick

**Simulationen** (im Originaltext vor- und nachahmende Spiele genannt), der Spieler tritt aus dem gewöhnlichen Leben heraus und schafft eine Welt mit eigenen Ordnungen

Weiterhin betont Jünger den freiwilligen Aspekt von Spielen: Ein potentieller Mitspieler muss die Freiheit haben das Spiel zu unterlassen oder sich seinen Regeln zu fügen. Die soeben genannten Regeln sind ein weiterer wichtiger Aspekt. Sie dürfen den Ablauf des

Spiels nur in gewissen Grenzen festlegen. Sind die Regeln zu streng definiert, so wird der Spielablauf zu stark begrenzt und lässt den Mitspielern keine Möglichkeiten zum Agieren im Spiel offen. Und gerade diese Unbestimmtheit ist der Spielraum, der zum Wesen des Spiels gehört und das Spiel kurzweilig macht.

Im Weiteren werden wir die einzelnen Kategorien näher betrachten.

### Glücksspiel

Von einem Glücksspiel reden wir dann, wenn wir dem Zufall die letzte Entscheidung überlassen. Den Zufall definieren wir hier als unerwartetes und nicht vorhersehbares Ereignis. Um ein Bestandteil des Spiels zu werden, muss der Zufall in den räumlichen und zeitlichen Grenzen erreichbar und wiederholbar sein. Außerdem muss der Ereignisraum $\Omega \neq \infty$ klar definiert und endlich sein. Ein Verfahren erzeugt die einzelnen Zufallsereignisse so, dass die atomaren Ereignisse nicht vorhersagbar sind. Unwichtig ist hierbei, durch wen das Verfahren angestoßen wird, ob durch einen Spieler, der würfelt oder Karten mischt und verteilt, oder jemanden, der nicht am Spiel teilnimmt, wie der Croupier am Roulette. Die Mitspieler müssen sich beim Agieren im Spiel auf die Möglichkeiten beschränken, die den Ausgang nicht in ihrem Sinne beeinflussen. Das spielentscheidende Ereignis wird ausschließlich vom Zufallsmechanismus bestimmt.

Beispiele für reine Glücksspiele sind:

**einfache Würfelspiele** - es kommt auf die höchste Zahl oder bestimmte Kombinationen an

**schwarzer Peter** - der Zufall bestimmt die Verteilung der Karten, das Zusammentreffen und Ablegen der zusammengehörigen Paare und den Spieler, der die letzte Karte in der Hand behält

**Roulette** - die angestoßene Kugel kommt nach mehreren Umläufen in einem Feld zum Stillstand und legt Gewinn oder Verlust fest

### Geschicklichkeitsspiel

Ein Spiel nennen wir Geschicklichkeitsspiel, wenn der Spieler durch seine körperlichen oder geistigen Fähigkeiten oder meist deren Kombination den Verlauf des Spiels beeinflusst und über Sieg oder Niederlage entscheidet. Besonders wichtig ist hier, dass die Entscheidungen nicht von Zufallsmechanismen getroffen werden sondern von den Mitspielern. Nach Rudolf Vogelsang [Vog63] gilt:

> „Jede Handlung unterliegt dem Anspruch auf Qualität. Wenn jemand beim Roulett-Spiel auf „ungerade“ setzt, ist das weder gut noch schlecht, der Zufall entscheidet über Gewinn oder Verlust. Wenn aber jemand beim Schachspiel eine Figur bewegt, so ist der Zug schwach oder stark. Das Urteil ist grundsätzlich sofort möglich, auch wenn es erst später offenbar wird.“

In einem Schachspiel beispielsweise lässt sich jeder Zug sofort nach dessen Ausführung bewerten. Auch wenn es für einen menschlichen Spieler unmöglich ist, sich alle weiteren Kombinationsmöglichkeiten vorzustellen und anhand derer sein Handeln zu bewerten, so ist eine Bewertung mit Hilfe von vollständigen Informationen trotzdem zu jeder Zeit möglich.

Beispiele für Geschicklichkeitsspiele sind:

**Fußball** - Kraft, Schnelligkeit, unmittelbares Auffassungsvermögen und Entschlossenheit sind Eigenschaften, die das Spiel beeinflussen

**Schach** - der körperliche Aufwand ist in Relation zum Geistigen verhältnismäßig klein

**Mikado** - benötigt werden hier Körperbeherrschung, vereinigt mit Geduld

#### Simulation

Als Simulation bezeichnen wir das Entfliehen aus dem realen Leben in eine Welt des Spiels, in der eine andere Gestalt angenommen werden kann und in der andere Regeln gelten. Zu dieser Gruppe von Spielen gehören nicht nur Spiele im klassischen Sinne, sondern auch künstlerische Darbietungen im Theater oder Verkleidungen beim Karneval. Aber auch das kleine Mädchen, dass mit ihrer Puppe spielt, simuliert eine Welt, in der sie die Mutter ist und sich um ihr Kind kümmern muss.

Selten kommt es allerdings vor, dass ein Spiel nur zu einer der drei vorgestellten Arten zählt. Viel öfter verbinden sich zwei der drei Eigenschaften zu einem Spiel. Die meisten Kartenspiele sind eine Vereinigung aus Glücksspiel und Geschicklichkeitsspiel. Die Zufallskomponente ist dann durch die zufällige Verteilung der Karten gegeben und mit Hilfe von Geschicklichkeit kann trotzdem ein Spieler mit schlechteren Karten ein Spiel gewinnen. Poker nimmt unter diesen Spielen eine Sonderrolle ein, da zum erfolgreichen Spiel ein skrupelloses Irreführen (Bluffen) des Gegners gehört. Auch viele andere Gesellschaftsspiele, wie beispielsweise „Mensch ärgere dich nicht“, kombinieren das zufällige Würfeln mit den persönlichen Entscheidungen, welche Figuren verrückt werden sollen.

Dazu kommt aber auch, dass derjenige, der ein Geschicklichkeitsspiel lernen möchte, einen besseren Spieler nachahmen kann, also simuliert. Derjenige, der eine Simulation spielt, muss auf der anderen Seite auch ein gewisses Geschick aufbringen, um zu gewinnen.

## F.2 Spieltheorie - mathematische Theorie von Spielen

Die mathematische Theorie der Spiele befasst sich mit der Möglichkeit, Spiele aus der realen Welt zu abstrahieren und dann mit mathematischen Methoden zu erfassen. Da sich aber ästhetische Gesichtspunkte sowie theologische oder pädagogische Aspekte einer formalen Analyse weitgehend entziehen, können nicht alle Spiele untersucht werden. Oft beschränkt man sich auf die Untersuchung von strategischen Spielen, die entweder auf Runden basieren oder durch eine Folge von Zügen beschrieben werden und eine starke Verwandschaft mit militärischen Konflikten aufweisen. Weiterhin wird unterteilt in

- Spiele mit endlich vielen Strategien
- Spiele mit unendlich vielen Strategien
- Duelle

womit wir uns im Folgenden auseinander setzen werden und den Begriff der Strategie näher erläutern.

### F.2.1 Spiele mit endlich vielen Strategien

Die meisten Spiele lassen sich als eine Folge von Zügen beschreiben. Diese Züge können entweder nur von den persönlichen Entscheidungen der Spieler abhängen oder auch als Zufallszüge durch ein Zufallsereignis - Würfeln oder Werfen einer Münze - bestimmt werden. Wir können die Beschreibung des Spiels vereinfachen, indem wir den Begriff der Strategie einführen. Bei der Durchführung einer Partie des Spiels kann jeder Spieler, anstatt seine Entscheidung bei jedem Zug zu treffen, im Voraus einen vollständigen

Spielplan unter Berücksichtigung aller Möglichkeiten aufstellen. Einen solchen Plan nennen wir eine Strategie. Er umfasst die dem Spieler gemäß den Spielregeln zugängliche Information. Somit geht durch die Verwendung einer Strategie keine Handlungsfreiheit verloren, da die Strategie bei der Festlegung der Aktionen des Spielers die zur Verfügung stehende Information berücksichtigt.

Jede Art, in der ein bestimmter Spieler ein Spiel vom Anfang bis zum Ende durchführen kann, stellt eine Strategie des Spielers dar. Werden all diese verschiedenen Arten aufgezählt, so erhalten wir die Gesamtheit der Strategien des Spielers. Natürlich ist es möglich, dass der Spieler einige dieser Strategien niemals anwendet. Trotzdem sind sie in der Gesamtheit enthalten. Die Kombinationen aller Strategien aller Spieler liefern sämtliche Möglichkeiten des Spielverlaufs.

Jedes Spiel, das ursprünglich durch eigenverantwortliches Ziehen oder zufallbasiertes Ziehen festgelegt wird, lässt sich auch mit Hilfe von Strategien beschreiben. Aus dieser Sicht besteht ein Spiel für jeden Spieler aus der Wahl der Strategie, wobei er nur die Regeln des Spiels kennt. Folglich führt die Strategie das Spiel durch.

Ein grundlegendes Problem der Theorie strategischer Spiele besteht in der Bestimmung der besten Strategie für jeden Spieler.
Das Spiel ist genau dann endlich, wenn jeder Spieler nur endlich viele Strategien zur Auswahl hat. Die Methoden zur Lösung endlicher Spiele unterteilen sich in:

- raten und verifizieren - es wird eine Strategie ausgewählt und überprüft, ob sie erfolgreich ist
- prüfen der Teilstrategien - das Gesamtproblem wird in Teilprobleme zerlegt und es wird geprüft, ob die Konkatenation der Teillösungen eine Gesamtlösung darstellt
- sukzessive Approximationen - das Gesamtproblem wird in Teilprobleme erlegt, für die optimale Lösungen gesucht werden

### F.2.2 Spiele mit unendlich vielen Strategien

Bei den Spielen mit endlich vielen Strategien wählt jeder Spieler seine Strategie aus einer endlichen Menge aus. Wie beim Schach kann die Anzahl der Strategien sehr groß werden, trotzdem ist sie endlich. Indem wir die Anzahl der Lösungen als Menge betrachten, und nur ein Intervall aus dieser Menge als Lösungen zulassen, können wir von endlichen zu unendlichen Strategien abstrahieren, indem wir die Grundmenge der Lösungen ändern. Wäre die Grundmenge bei einem Schachspiel die Menge der natürlichen Zahlen, also abzählbar unendlich, ein beliebiges Intervall daraus folglich abzählbar, so können wir auf dem gleichen Intervall die Menge der reellen Zahlen betrachten und erhalten so eine Menge mit unendlich vielen Elementen. Wir können also verallgemeinern, in dem wir auch solche Spiele betrachten, bei denen jedem Spieler unendlich viele Strategien über einem abgeschlossenen Intervall zur Verfügung stehen. Speziell können wir annehmen, dass jeder Spieler eine Menge von Strategien hat, aus der er für die Durchführung des Spiels eine Strategie auswählen kann.

### F.2.3 Duelle

Die Spieltheorie lässt sich zur Lösung einer großen Klasse von Zeitwahlproblemen verwenden. Bei diesen Problemen sind die Aktionen der Spieler von Vornherein festgelegt im Gegensatz zum Zeitpunkt der Aktionen. Folglich besteht ein Interessenkonflikt, denn jeder Spieler möchte seine Aktion so lange wie möglich geheim halten, wird aber bestraft, wenn er zu lange wartet. Diese Form von Zeitspielen werden Duell genannt.

## F.3 Vom analogen zum digitalen Spiel

Wie wir in der Einleitung bereits erwähnt haben, liegt es in der Natur des Menschen zu spielen. Nach den anfänglich eher technischen Versuchen an Universitäten in den 50er Jahren wuchsen die Kapazitäten von Computern. Als erstes Videospiel wird meist das vom Physiker William Higinbotham geschriebene „Tennis for Two“ angesehen, das aus einem Analogcomputer und einem Oszillographen bestand.

Mit der stetigen Weiterentwicklung der Computer wurden auch immer aufwendigere grafische Simulationen erschaffen. In den 70er Jahren wurden aus den bestehenden massenproduzierten Fernsehgeräten relativ preisgünstige Spielautomaten entwickelten, die ab dann auch der Öffentlichkeit zur Verfügung gestellt wurden. Im gleichen Jahrzehnt entwickelten Firmen wie Atari Videospielekonsolen für den Heimanwender. In den folgenden zwei Jahrzehnten wuchs die Spieleindustrie immer weiter durch die Entwicklung immer schnellerer Computer und auch besserer Konsolen, so dass Videospiele im 21. Jahrhundert eine der einflussreichsten Freizeitgestaltungsformen in wirtschaftlich entwickelten Staaten sind.

Heutzutage überschreiten die Umsätze der Videospielindustrie die der Filmindustrie und der Einfluss der Videospielindustrie auf junge Menschen ist vergleichbar groß wie der der Filmindustrie.

## F.4 Forschungsinhalte

Im Fokus der Forschung stehen digitale Spiele. Der Grund für die Wahl des Computers ist darin begründet, dass Computer nicht nur beliebige traditionelle Spiele wie Brett- oder Kartenspiele darstellen können, sondern auch eine neue, bis dahin nicht gekannte Form von Spielen möglich machen, die Videospiele. Diese kombinieren Inhalte und Techniken von narrativen Komponenten mit traditionellen Spielen und nutzen die akustischen und visuellen Möglichkeiten, die Computer bieten.

### F.4.1 Geschichte des Spiels

Spielen stellt das älteste bekannte Kulturphänomen überhaupt dar. Es ist nicht nur älter als Sprache, Schrift und Kunst, sondern sogar älter als der homo sapiens selbst, denn auch viele Tiere spielen und nutzen das Spiel zur (Meta-)Kommunikation untereinander und mit dem Menschen. Aus diesem Grund führte Huizinga 1939 in [Hui94] den Begriff des homo ludens (lat. der spielende Mensch) ein. Der homo ludens entwickelt sich selbst durch die von ihm gemachten Erfahrungen im Spiel, da er in diesem eine gewisse Handlungsfreiheit hat und selber denken muss um Probleme zu lösen. Huizinga ging davon aus, dass unsere heutigen kulturellen Systeme wie Politik, Ökonomie, Religion und Justiz sich ursprünglich aus spielerischen Verhaltensweisen entwickelt haben.

Die Bedeutung des Spiels für den Menschen wird verständlicher, wenn wir daran denken, dass jeder Mensch sich von Geburt an mit Spielen beschäftigt und Spiele die Entwicklung vom Kind zum Erwachsenen maßgeblich beeinflussen. Aber auch im erwachsenen Alter nimmt das Spiel einen gewissen Bereich im Leben vieler Menschen ein: viele praktizieren in ihrer Freizeit verschiedenste Sportarten, die alle unter den Begriff Spiele fallen.

Obwohl das Spielen schon seit Menschengedenken zum Alltag gehört, befasste sich erst Mitte des 20. Jahrhunderts der oben genannte Kulturanthropologe Huizinga als erster wissenschaftlich mit diesem Thema.

Nach Huizinga setzten sich noch Ludwig Wittgenstein (1950), Roger Caillois (1961) und Gregory Bateson (1972) mit Spielen auseinander.

Erst gegen Ende der 1990er Jahre begannen sich Veröffentlichungen zur Thematik zu häufen, und es formierte sich die Basis einer neuen wissenschaftlichen Disziplin. Dies war maßgeblich der zunehmenden Etablierung digitaler Spiele, ihren Auswirkungen auf die Gegenwartskultur, sowie ihrer wachsenden wirtschaftlichen Bedeutung zu verdanken. Die Bezeichnung „Ludologie“ wurde 1999 durch einen Artikel von Gonzalo Frasca einem breiteren Fachpublikum bekannt und in der Folge zusehends akzeptiert.

### F.4.2 Definition des Spielbegriffs

Das Spiel (vom althochdt. spil = Tanzbewegung) wird definiert als Tätigkeit, die ohne bewussten Zweck zum Vergnügen, zur Entspannung und allein aus Freude an ihrer Ausübung betrieben wird. Der Begriff Spiel trägt umgangssprachlich verschiedenste Semantiken. So spielt sowohl ein Musiker auf einem Instrument, ein Kind mit einem Ball, ein Schauspieler seine Rolle oder ein Spieletester ein Computerspiel. Ziel der Forschung in diesem Bereich ist eine Abgrenzung und Definition des Spielbegriffs. Es sollen nicht nur verschiedene Spieletypen kategorisiert werden, sondern auch beschrieben werden, was zu Spielen gezählt werden soll.

### F.4.3 Terminologie zur Beschreibung und Klassifizierung von Spielen und Spielgenres

Computerspiele versuchen teilweise die Realität auf dem Computer abzubilden. In vielen Spielen wird die Realität allerdings verfremdet, beschnitten oder erweitert, um den Spaß am Spiel zu vergrößern. Je nach repräsentiertem Ausschnitt ist es möglich, das Spiel in eine Klasse einzuteilen. Durch immer größere Hardwareressourcen lassen sich auch immer detailliertere oder größere Ausschnitte darstellen und somit wird eine Klassifikation in nur ein Genre immer schwerer.

Spiele lassen sich grob unterteilen in:

- Actionspiele (Beat 'em up, Ego-Shooter, Jump'n'Run, Shoot'em up, ..)
- Abenteuerspiele (Adventures, Rollenspiele, ..)
- Strategiespiele (Aufbaustrategie, Echtzeitstrategie, rundenbasiertes Strategiespiel, ..)
- Simulationen (Flugzeugsimulationen, Lebensimulationen, Autosimulationen, ..)
- Sportspiele (Rennspiel, Rennsimulation, ..)
- Sonstige (Retrospiel, Puzzle, Lernspiele, ..)

### F.4.4 Regeln, Spielmechanik und Gameplay

Den Ablauf eines Spiels bezeichnen wir als Spielmechanik und definieren damit auch die Spielregeln. Das Spielerlebnis wird maßgeblich durch diese Definition geprägt. Durch eine Anfangssituation und die Möglichkeiten, die dem Spieler geboten werden, leiten wir das Spiel in erwünschte Bahnen. Gleichzeitig nehmen wir dadurch dem Spieler die Möglichkeit beliebige Aktionen auszuführen.

Zum Ablauf des Spiels und den Regeln gehören aber nicht nur Vorschriften für die erwünschten und unerwünschten Aktionen des Spielers, sondern auch die Definition der Spielwelt, gegebenenfalls der Gegner und natürlich auch der Ziele, die ein Spieler erreichen soll.

Das Gameplay definieren wir als die Interaktion zwischen Spieler und Spiel. Häufig wird dieser Begriff von Autoren für die Bewertung eines Spiels herangezogen.

### F.4.5 Cybertext und ergodische Literatur, interaktive Erzählformen, Storytelling

Espen Aarseth [Aar97] beschreibt das Grundkonzept der ergodischen Literatur wie folgt: Der Spieler muss nicht nur einen Text lesen, sondern auch Bewegungen ausführen. Der Spieler ist nicht nur passiver Leser, wie der Leser eines Buchs, sondern aktiv, indem er die Geschichte durch seine eigenen Aktivitäten unter seiner Kontrolle weiterführt. Er bestimmt die Reihenfolge, in der der Text gelesen wird, bewertet wichtige Textpassagen und lässt gegebenenfalls Unwichtige weg. Aus diesem Grund unterscheiden sich Spiele von Texten.

Cybertexte sind eine Unterkategorie der ergodischen Literatur und zeichnen sich durch die Produktion der Texte durch maschinelle Berechnung zur Laufzeit aus. Im Gegensatz zur ergodischen Literatur darf der Text also nicht schon vor dem Start des Spiels feststehen, sondern muss sich aus den Aktionen des Spielers generieren. Demzufolge sind digitale Spiele im Computer Cybertexte, da sie diese Eigenschaften haben. Zur Laufzeit wird z.B. die künstliche Intelligenz berechnet oder physikalische Eigenschaften oder Verhalten von Computergegnern. Der Inhalt, durch den ein Spiel repräsentiert wird, ergibt sich also erst durch die komplexen Aktionen des Spielers, der ergodischen Textauswahl und der Berechnungen des Spiels bezüglich KI, CI, Umwelt und Spielgeschehen.

### F.4.6 Spiel als Simulation vs. Spiel als Narration

Monika Flundernik [Flu06] definiert die Erzähltheorie:

> „Die Erzähltheorie, oder mit dem international anerkannten Terminus: die Narratologie [..], ist die Wissenschaft vom Erzählen. Es handelt sich hierbei um die Untersuchung der Erzählung als Gattung mit dem Ziel, ihre typischen Konstanten, Variablen und Kombinationen zu beschreiben und innerhalb von theoretischen Modellen (Typologien) die Zusammenhänge zwischen den Eigenschaften narrativer Texte zu klären [..]“

Nachdem wir den Begriff der Simulation bereits im Kapitel F.1.2 auf Seite 122 besprochen haben ist verständlich, weshalb an dieser Stelle Bedarf nach Forschung und Klärung besteht. Hier versucht die Ludologie zu klären, wie Spiele aufgefasst werden sollten.

### F.4.7 Spieleentwicklung

Die Spieleentwicklung (eng. Game Design) befasst sich mit der Erzeugung der Spielvision sowie der Spielwelt, also den Objekten und Regeln im Spiel. Es handelt sich je nach Spiel um eine Kombination aus Narration und Simulation. Aus dem Prozess der Spieleentwicklung entsteht ein Schriftstück in dem die Konzepte des Spiels festgehalten werden. Obwohl die Spieleentwicklung eher theoretisch Eigenschaften des späteren Spiels betrachtet, muss im Vorfeld bereits daran gedacht werden, auf welchem System die Software laufen soll. Die verschiedenen Systeme, ob Spielekonsole, Computer oder Mobiltelefon haben verschiedenste Anforderungen und Möglichkeiten der Darstellung und Datenverarbeitung, weshalb schon in diesem frühen Anfangsstadium der Planung wesentliches Augenmerk darauf gelegt werden muss.

### F.4.8 Immersion, Immersionsebenen, Flow-Theorie

In einer virtuellen Welt beschreibt der Begriff Immersion das Eintauchen in die künstliche Welt. Das damit verbundene Gefühlt wird als Flow bezeichnet und mit der vom Psychologen Mihaly Csikszentmihalyi vorgestellten Flow-Theorie (vgl. [Csi05]) beschrieben. In [Bar03] unterteilt Richard Bartle Immersion in 4 Ebenen:

**player** - die Spielfigur beeinflusst die Spielwelt

**avatar** - die Spielfigur repräsentiert den Spieler in der Spielwelt

**character** - der Spieler identifiziert sich mit der Spielfigur

**persona** - die Spielfigur ist Teil der Identität des Spielers; allerdings spielt der Spieler keine Figur in einer virtuellen Welt sondern befindet sich selbst in der virtuellen Welt

Csikszentmihalyi beschreibt den Flow weiter durch folgende Eigenschaften, die allerdings nicht alle zutreffen müssen: Der Spieler ist der auszuübenden Aktivität gewachsen, er kann sich auf seine Aufgaben im Spiel konzentrieren. Seine Aktivitäten haben eindeutige Ziele und sofortige Rückmeldungen. Der Spieler glaubt seine Aktivitäten im Spiel zu kontrollieren und vergisst dabei seine Sorgen und die Zeit in der realen Welt.

Der Flow liegt dabei zwischen einer Unter- und einer Überforderung des Spielers im Spiel, bei dem die Anforderungen des Spiels an den Spieler den Fähigkeiten des Spielers gleich kommen.

### F.4.9 Entwicklungsmuster von Charakteren in persistenten Spielwelten

Persistente Welten sind virtuelle Welten, welche Teile der realen Welt simulieren und die sich dadurch auszeichnen, dass sie von einem Spieler zu jeder Zeit betreten und verlassen werden können. Wichtiger Aspekt der persistenten Welten ist auch die kontinuierlich verlaufende Zeit. Ziel der Spiele, in denen persistente Spielwelten existieren, ist das Spielen mit einem Charakter (siehe Immersionsebenen in F.4.8 auf Seite 127) um diesen zu verbessern. Bekannteste persistente Welt ist momentan Azeroth in World of Warcraft.

Meist existieren persistente Welten in Online Spielen, damit die simulierte Welt durch die Teilnahme von anderen Spielern als realer empfunden wird. Allerdings sind auch persistente Welten in Offline Spielen denkbar. Bei dem erneuten Betreten eines Spielers in einem Offline Spiel müssen allerdings die verstrichene Zeit, sowie die vergangenen Aktionen simuliert werden, bevor der Spieler die persistente Welt betritt.

### F.4.10 Lernvorgänge in Spielen, Lernspiele

In der Pädagogik werden Lernspiele verwendet, um erwünschte Lern- und Übungseffekte beim Spieler hervorzurufen bzw. zu fördern. Spielepädagogen sind sich einig, dass sämtliche Spiele Lern- und Übungseffekte hervorrufen. Allerdings gibt es nicht nur erwünschte sondern auch unerwünschte Effekte, welche sich zum Beispiel in einer Spielsucht manifestieren können. Seit der Verbreitung von Computern und Spielekonsolen versuchen Spieleindustrie und Schulbuchverlage Lernspiele in digitaler Form auf den Mark zu bringen. Zielgruppe sind zwar vorwiegende Kinder und Jugendliche, trotzdem gibt es auch Lernsoftware z.B. für das Lernen von Fremdsprachen für Erwachsene. Durch den spielerischen Charakter der Lernspiele soll die Motivation des Spielers gefördert werden.

### F.4.11 Visualisierung virtueller Welten

Ein weiterer wichtiger Aspekt der Forschung ist die Visualisierung von virtuellen Welten. Computer beschränken sich generell auf die Darstellung auf dem Display. Um ein Gefühl der Immersion (vgl. F.4.8 auf Seite 127) zu generieren reicht dies aber nicht immer aus. Aus diesem Grund wird spezielle Hardware in Form von Head-Mounted-Display, Großbildleinwänden oder CAVE (ein Raum zur Projektion dreidimensionaler Illusionswelt) verwendet und erforscht.

### F.4.12 Künstliche Intelligenz in Computerspielen

Um Spiele realistischer zu gestalten, also die Simulation mit möglichst wenig Einschränkungen zu erzeugen, muss auf Aktionen von Spielern reagiert werden. Da der Spieler in der Spielwelt teilweise unendlich viele Möglichkeiten zum Agieren hat, kann nicht für jede Aktion eine Gegenaktion abgespeichert werden. Gelöst wird dieses Dilemma durch den Einsatz von Wahrscheinlichkeiten und vor allem auch künstlicher Intelligenz, die dann zum Beispiel durch das Erkennen von Eingaben des Spielers Gegner oder andere Objekte im Spiel steuert.
Die KI wird in 4 Teilgebiete unterteilt:

**visuelle Intelligenz** - durch Mustererkennung und Analyse kann beispielsweise eine Person in einem virtuellen Raum erkannt werden

**sprachliche Intelligenz** - Sprache eines Spielers kann analysiert werden um darauf zu reagieren und auch gegebenenfalls sprachlich darauf zu antworten

**rationale Intelligenz** - auf Expertensystemen basierendes Wissen, um auf Aktionen des Spielers zu reagieren

**emotionale Intelligenz** - wird manchmal auch verwendet und setzt sich aus den Bereichen „Selbstbewusstsein, Selbstmotivation, Selbststeuerung, soziale Kompetenz und Empathie“ zusammen (Stimmungsanalyse); wird auch zum Reagieren auf Aktionen des Spielers verwendet

### F.4.13 Mensch-Computer-Interaktion, Interface-Design

Die Forschung im Bereich Mensch-Computer-Interaktion vereint viele Aspekte der Ludologie. Zum Einen möchte man das Gefühl der Immersion (vgl. F.4.8 auf Seite 127) verbessern, zum Anderen auch die Interaktion des Spielers im Spiel erleichtern und näher an der Realität gestalten. Ein Beispiel für eine intuitivere Steuerung ist die *Wii Remote* für die Spielkonsole Wii von Nintendo. Indem der Schwerpunkt bei diesem Eingabegerät auf die Bewegung des Arms gelegt wird, verändert sich die Simulation der Aktion in eine natürliche Bewegung, wie man sie auch bei vergleichbaren realen Spielen erwarten würde.

### F.4.14 Wirkungsforschung, Gewalt in Computerspielen

Durch den Ansatz, die Realität so gut wie möglich zu simulieren, werden Computerspiele immer detaillierter. Diese Verbesserung fördert natürlich das Gameplay (vgl. F.4.4 auf Seite 125) und auch die Immersion (vgl. F.4.8 auf Seite 127). Diese Verbesserungen machen es aber auch immer schwieriger, den Unterschied zwischen einem abstrakten Spiel und einer Filmszene oder Nachrichtensendung zu sehen. Einzig und allein die Art der Aktion des Beteiligten unterscheidet verschiedene Medien und somit besteht die Gefahr, dass Grenzen aufweichen oder verschwinden. Die Forschung in diesem Bereich versucht

die Ursachen zu lokalisieren und die Zusammenhänge zwischen Gewalt in Spielen und in der Realität genau zu beleuchten.

### F.4.15 Sport und Spiel, elektronischer Sport

Nachdem sich aus den Freizeitbeschäftigungen Sport und Spiel ihre elektronischen Pendants entwickelt haben, gehen manche Länder (Brasilien, China) dazu über, dass die elektronischen Spiele (kurz E-Sport) nach und nach auch als Sportarten anerkannt werden. Der Deutsche Olympische Sportbund erkennt E-Sport noch nicht als Sportart an.

In Wettkämpfen, die an bestimmten Orten oder auch von Zuhause bestritten werden können, treten einzelne Spieler oder auch Gruppen von Spielern, so genannte Clans, gegeneinander an.

In Deutschland wird der E-Sport durch den deutschen eSport-Bund repräsentiert.

### F.4.16 Communities und soziale Interaktion in Computerspielen, Videospielkultur

Durch die Tatsache, dass viele Computerspiele die Möglichkeit anbieten, mit anderen Spielern ein Spiel gemeinsam oder auch gegen andere Spieler auszuüben, entwickeln sich Gemeinschaften von Spielern, die soziale Interaktionen innerhalb der Spiele oder auch außerhalb der Spiele in Foren pflegen.

## F.5 Zusammenfassung

In dieser Ausarbeitung haben wir uns mit dem Phänomen Spiele beschäftigt. Von der Begriffserklärung der Ludologie sind wir dazu übergegangen zu definieren, was ein Spiel ist und in welche Kategorien man Spiele einteilen kann. Wir haben die mathematische Theorie von Spielen angeschnitten und haben uns dann mit den Forschungsinhalten der Ludologie auseinandergesetzt. Dabei haben wir uns um verschiedenste Aspekte (geschichtlich, sprachtheoretische, Informatik betreffend, psychologisch, pädagogisch, sportlich, ..) von Spielen gekümmert. Anschließend haben wir die Entwicklung vom analogen zum digitalen Spiel untersucht.

Wir haben festgestellt, dass viele dieser Themen sich nicht separat analysieren lassen, da es sehr viele Überschneidungen gibt. Außerdem ist das Thema Ludologie sehr vielseitig, so dass es sich nicht vollständig in einer solchen Ausarbeitung abhandeln lässt. Für weitere Informationen sei auf die Literaturliste verwiesen.

# G Maschinelles Lernen/Regelbasierte Steuerung

Es wird vorerst definiert, was hier unter Lernen im allgemeinen und „maschinellem Lernen“ im Speziellen zu verstehen ist.

Maschinelle Lernverfahren können grob in überwachte und unüberwachte Lernverfahren unterteilt werden. Es werden für beide Klassen Vertreter besprochen, wobei ein Schwerpunkt auf den überwachten Lernverfahren liegt.

Zum Abschluss werden Ideen skizziert, wie man mit Hilfe der vorgestellten Verfahren Nicht-Spieler-Charaktere (NSC) entwickeln könnte.

## G.1 Einleitung

### G.1.1 Was bedeutet Lernen?

Um zu Verstehen was „Maschinelles Lernen“ bedeutet, muss vorerst klar sein was wir unter Lernen im Allgemeinen verstehen. Es gibt viele verschiedene Definitionen vom Begriffs des Lernens, manche schließen einander aus, andere wiederum fassen den Begriff enger/weiter.

Das es von bestimmten Sachverhalten in der KI verschiedene – teilweise widersprüchliche – Auffassungen gibt, ist für die KI recht typisch. Ursache hierfür sind die verschiedenen wissenschaftlichen Disziplinen die sich mit KI beschaffen. So arbeiten hier Psychologen, Biologen, Soziologen, Statistiker und weitere an Methoden der künstlichen Intelligenz. Die unterschiedlichen Ausgangspunkte von diesen, machen eine Verständigung häufig recht schwer.

Fangen wir mit einer Definition eines der einflussreichsten Sozialwissenschaftler [Sim83, Simon 1983]an:

> Lernen ist jeder Vorgang, der ein System in die Lage versetzt, bei der zukünftigen Bearbeitung derselben oder einer ähnlichen Aufgabe diese besser zu erledigen.

Diese Definition scheint treffend zu sein. Indes hat sie mindestens ein Problem, dass an folgendem Beispiel illustriert wird:

Nehmen wir die Aufgabe zu Lernen möglichst effektiv einkaufen zu gehen. Der Definition von Simon zu folge, dürften wir genau die Erkenntnisgewinne als Lernen betrachten, die uns dazu befähigen schneller und günstiger einkaufen zu gehen. Das ist aber nicht das, was von einem Großteil der Wissenschaft unter Lernen verstanden wir. Lernen wäre danach – in unserem Beispiel – auch alles, was wir beim Einkaufen so nebenbei aufnehmen und verarbeiten. Also z.B. welche Geschäfte und Einrichtungen auf dem Weg zu unserem Geschäft liegen. Diese Erkenntnisse sind zwar nicht geeignet schneller und günstiger einzukaufen, aber stellen doch Lernen dar. Es bleiben neue Erkenntnisse zurück, auf die in anderen Situationen gewinnbringend zurückgegriffen werden kann.

Kommen wir zu einer weiteren populären Definition [Mic86, Michalski 1986]:

> Lernen ist das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.

Diese Definition hat nicht das Problem der Ersten, ist jedoch wieder so allgemein, dass sie kaum noch nützlich ist. Andererseits wird sie zudem als zu eng kritisiert, da beim Lernen nicht immer Repräsentationen im Spiel sein müssen.

Zum Schluss muss noch auf die Definition des Lernens über die Rückführung auf die drei logischen Schlussformen erwähnt werden[GR00]:

**Deduktion:** Aus „Alle Menschen sind sterblich." und „Sokrates ist ein Mensch." schließe „Sokrates ist sterblich." (wahrheitserhaltend)

**Induktion:** Aus „Sokrates ist ein Mensch." und „Sokrates ist sterblich." schließe „Alle Menschen sind sterblich." (falschheitserhaltend bzgl. der zweiten Aussage)

**Abduktion:** Aus „Sokrates ist sterblich." und „Alle Menschen sind sterblich." schließe „Sokrates ist eine Mensch." (falschheitserhaltend bzgl. der ersten Aussage)

Lernen über diese Schlussformen zu definieren ist zu einschränkend, da große Bereiche vom (Maschinellen) Lernen mit dieser Art Schlüsse zu ziehen, nichts zu tun haben.

### G.1.2 Was bedeutet Maschinelles Lernen?

Im vorherigen Abschnitt haben wir gesehen, dass es nicht einfach ist, klar zu definieren was Lernen ist. Mit dem Maschinellen Lernen ist das einfacher. Es wird vorerst der Begriff der Lernaufgabe wie im [GR00, Görz] eingeführt:

> Eine Lernaufgabe wird definiert durch eine Beschreibung der dem lernenden System zur Verfügung stehenden Eingaben (ihrer Art, Verteilung, Eingabezeitpunkte, Darstellung und sonstigen Eigenschaften), der vom lernenden System erwarteten Ausgaben (ihrer Art, Funktion, Ausgabezeitpunkte, Darstellung und sonstigen Eigenschaften) und den Randbedingungen des Lernsystems selbst (z.B. maximale Laufzeiten oder Speicherverbrauch).

Maschinelles Lernen lässt sich so, einfach über die Menge aller Lernaufgaben definieren. Lernaufgaben unterscheiden sich im wesentlichen über den Umfang des Feedbacks in der Lernphase. Zum einen gibt es das überwachte Lernen.

Hier bekommt das System eine Menge bereits klassifizierter Beispiele als Eingabe.

Beim unüberwachten Lernen fehlt diese Klassifizierung und das System muss selbständig Klassenzugehörigkeiten erkennen. In den folgenden Abschnitten werden Vertreter dieser beiden Klassen vorgestellt und damit der Unterschied klarer.

Letztlich soll auch noch eine Klasse von Lern-Algorithmen angesprochen werden: Wenn der Algorithmus für Entscheidungen belohnt oder bestraft wird, spricht man von vom Bekräftigungslernen (Reinforcement-lernen). Auf diese Klasse von Lernalgorithmen wird in dieser Ausarbeitung nicht näher eingegangen.

## G.2 Vorstellung ausgewählter Verfahren

### G.2.1 Überwachtes Lernen

Hier werden drei klassische Vertreter für das überwachte Lernen vorgestellt.

Bei den ersten beiden Lernverfahren handelt es sich um Methoden für das Funktionslernen aus Beispielen. Dies ist mit die populärste Lernaufgabe.

Mit den Assoziationsregeln wird die populärste Methode aus dem Data-Mining behandelt.

**Tabelle G.1:** Beispielpflanzendaten

| ID | Feuchte | Säure | Temp. | Klasse |
|---|---|---|---|---|
| 1 | trocken | basisch | 7 | W |
| 2 | feucht | neutral | 8 | N |
| 3 | trocken | neutral | 7 | W |
| 4 | feucht | alkalisch | 5 | N |
| 5 | trocken | neutral | 8 | N |
| 6 | trocken | neutral | 6 | W |
| 7 | trocken | neutral | 11 | N |
| 8 | trocken | neutral | 9 | N |
| 9 | trocken | alkalisch | 9 | W |
| 10 | trocken | alkalisch | 8 | W |
| 11 | feucht | basisch | 7 | N |
| 12 | feucht | neutral | 10 | W |
| 13 | trocken | basisch | 6 | W |
| 14 | feucht | alkalisch | 7 | N |
| 15 | trocken | basisch | 3 | N |
| 16 | trocken | basisch | 4 | W |

### Entscheidungsbaumlernen

Da sich Entscheidungsbäume [Breiman et al., 1984; Quinlan, 1993; Murthy, 1998] auch als Regelwerk aufzufassen lassen und in dieser Ausarbeitung auch noch was zur regelbasierten Steuerung gesagt werden soll, wird das Entscheidungsbaumlernen vergleichsweise tief behandelt.

### Einführung des Beispiels

Das Beispiel dazu stammt aus der Pflanzenökologie [KWDD98] und befasst sich mit der Frage, wie man eine Funktion bestimmen kann, die in Abhängigkeit der Parameter Bodenfeuchte, Säuregehalt und Temperatur angibt, ob eine bestimmte Pflanzenart dort wachsen kann.

Wie könnte man vorgehen? Eine in der Praxis durchgeführte Methode ist es, dass man so durch die Wälder und Felder streift, die entsprechenden Werte aufnimmt und notiert welche Pflanzen an den Messstellen wachsen. Wenn man dies tut, dann kann z.B. eine Tabelle wie Tabelle G.1 dabei herauskommen.

Die Spalte „Klasse“ gibt an, ob die entsprechende Pflanze dort wachsen kann, oder nicht. Hier steht ein „W“ für ein „wächst an diesem Standort“ und ein „N“ für das Gegenteil.

Nun haben wir einige Werte, von denen wir wissen, dass sie aus der Realität kommen. Wir wollen nun daraus eine Funktion ableiten, die bei Gabe von einem freiwählbaren Tripple aus Feuchte, Säure und Temperatur uns eine möglichst gute Prognose liefert, ob dort die von uns gerade betrachtete Pflanze wachsen kann oder nicht.

Bei diesem Lernen gibt es verschiedene grundlegende Dinge, die es zu beachten gilt. Im folgenden wird das größte Problem vorgestellt. Später, wenn das Entscheidungsbaumlernen besprochen wird, werden Konzepte vorgestellt wie man dem genannten Problem begegnen kann.

**Trainingsfehler:** Wir unterstellen, dass wir eine Funktion $h$ gefunden haben, die uns die gewünschten Prognosen liefert. Dann verstehen wir unter dem Trainingsfehler die Summe aller Abweichungen, die diese Funktion von den Eingabeinstanzen hat.

Der nächste Begriff ist wichtiger und setzt auf dem Ersten auf. Es geht um den sogenannten „wahren Fehler".

**Der wahre Fehler:** Der wahre Fehler gibt an, wie gut unsere Funktion auf neuen, noch unbekannten Anfragen ist. Hier ist zu berücksichtigen, dass nicht alle theoretisch vorstellbaren Kombinationen aus Feuchte, Säuregehalt und Temperatur gleichwahrscheinlich sind. Wir definieren $D$ als die Wahrscheinlichkeitsverteilung aller Eingaben $X$ die die Auswahlwahrscheinlichkeit bestimmter Eingaben aus der Menge aller Eingaben angibt. Dann ist der wahre Fehler definiert als:

$$error_D(h, f) = E_D[error(h(x), f(x)]$$

Hier ist $f(x)$ die wirkliche Funktion, die wir mit $h(x)$ versuchen zu approximieren. $x$ ist ein Eingabetupel. $error(h(x), f(x))$ gibt den Fehler an, den unsere Funktion $h(x)$ im Vergleich zu der wirklichen Funktion $f(x)$ hat. $E_D$ gibt dann den erwarten Fehler an den unsere Funktion über allen Eingaben nach der erwarteten Verteilung $D$ hat.

Da $D$ und $f$ unbekannt sind, kann der wahre Fehler nicht bestimmt werden. Wir werden später Möglichkeiten kennen lernen um den wahren Fehler anzunähern.

Das Hauptproblem ist, dass wir Gefahr laufen unsere Funktion $h(x)$ zu sehr an unsere Trainingsdaten anzupassen. Da die Trainingsdaten mit hoher Wahrscheinlichkeit nicht der Verteilung $D$ entsprechend bestimmt sind, besteht die Gefahr dass wir aus diesem Grund den Wahren Fehler vergrößern.

### Der Aufbau des Baumes

Kommen wir nun zurück zur Methode des Entscheidungsbaumlernens. Diese nimmt die gesammelten Daten aus unserem Beispiel auf und berechnet daraus folgenden Baum: (Abbildung 52)

Um einen Klassifizierung vorzunehmen, kann dieser Baum dann einfach von der Wurzel aus durchlaufen werden. An den Knoten wird sich – abhängig von der Ausprägung der Attribute – für den entsprechenden Nachfolgeknoten entschieden. Dies wird wiederholt bis man an einem Blatt ankommt. Dieses Blatt enthält dann die Klasse als Wert.

Damit ist die Laufzeit direkt von der Anzahl der Attribute abhängig und entspricht der Tiefe des Baumes.

Ein Algorithmus um so einen Baum aufzubauen geht folgendermaßen vor:

1. Initialisiere die Menge $Q$ mit der Menge aller Attribute
2. Bestimme dann für jedes Attribut den Trainingsfehler, den wir hätten wenn wir nur anhand dieses Attributs klassifizieren würden.
3. Nehme das Attribut, mit dem geringsten Trainingsfehler und hänge es an die Blätter. Wenn es keine Blätter gibt, dann wird dieser Attributknoten zum

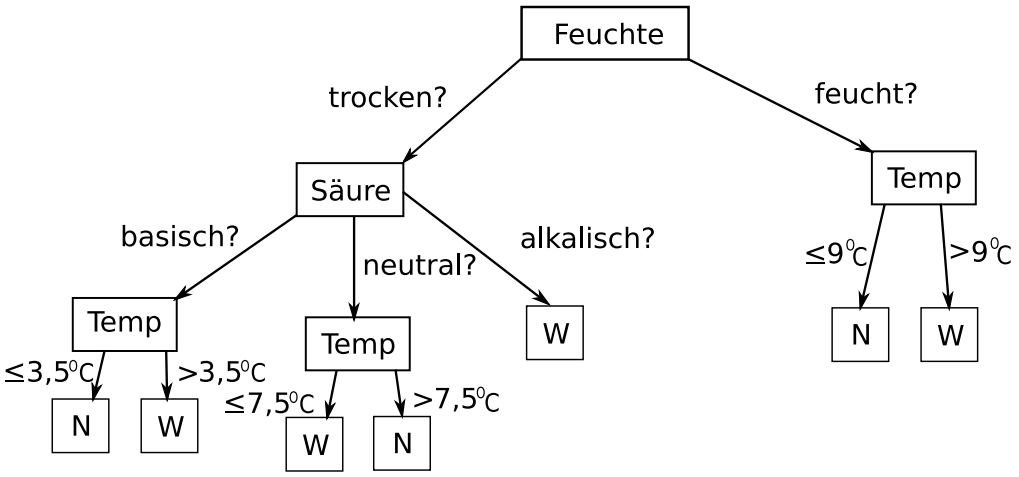



**Abbildung 52:** Der komplette Entscheidungbaum

**Tabelle G.2:** Trockene Böden

| Id | K | M |
|---|---|---|
| 1 | W | W |
| 3 | W | W |
| 5 | N | W |
| 6 | W | W |
| 7 | N | W |
| 8 | N | W |
| 9 | W | W |
| 10 | W | W |
| 13 | W | W |
| 15 | N | W |
| 16 | W | W |

4. Wiederhohle 2. und 3. solange bis die Menge $Q$ leer ist.

Da alle Schritte bis auf den 2. trivial sind, wird hier nur der 2. Schritt anhand unseres Standardbeispiels erläutert.

**2. Schritt:** Bestimmen des Attributs mit dem geringsten Trainingsfehler

Wir betrachten alle noch verbliebenen Attribute und bestimmen für jedes den Trainingsfehler.

**Betrachtung nach Feuchte:** Wir partitionieren die Eingabe nach ihren Werten für die Bodenfeuchte. Da hier nur zwei Werte zulässig sind, kommen dabei zwei Teilmengen raus. Wir haben insgesamt 16 Messwerte, davon waren 11 Böden trocken, die entsprechenden Standorte sind in Tabelle G.2 aufgeführt.

Die 1. Spalte referenziert den Datensatz. Die 2. Spalte gibt das „Ergebnis“ an, ob die betrachtete Pflanze an dem untersuchten Ort wuchs oder nicht. Wir stellen fest, dass sie bei trockenen Böden häufiger wuchs als nicht und damit ist dies unsere „Mehrheitsklasse“. Wir vermerken dies in der 3. Spalte.

Wenn wir immer, wenn wir einen trockenen Boden messen, voraussagen würden dass die betrachte Pflanze dort wächst, dann würde wir – gemessen an unseren Eingabedaten – 4 mal falsch liegen und 7 mal richtig. Die Fehlerrate liegt hier somit bei 4/11.

**Tabelle G.3:** Feuchte Böden

| Id | K | M |
|---|---|---|
| 2 | N | N |
| 4 | N | N |
| 11 | N | N |
| 12 | W | N |
| 14 | N | N |

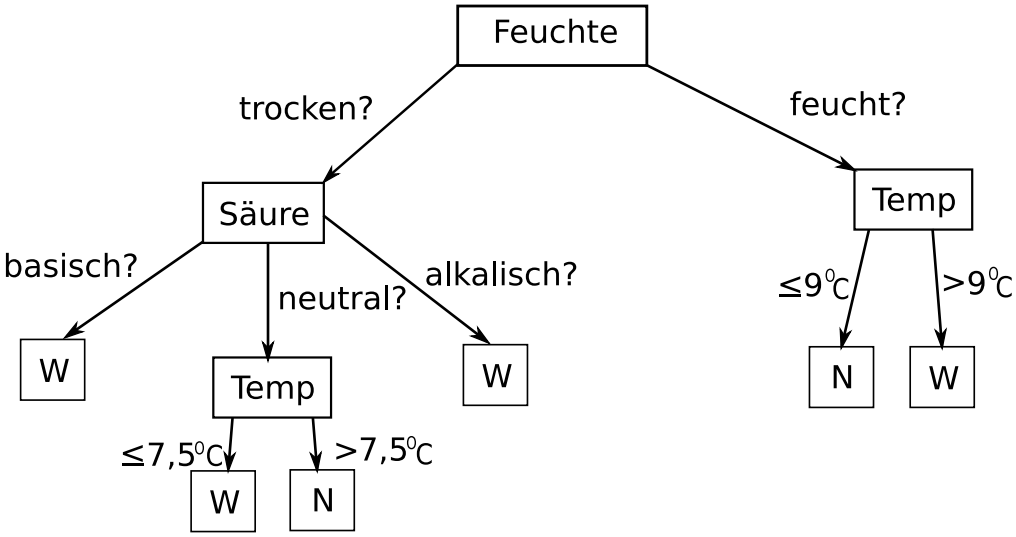



**Abbildung 53:** Der gestutzte Baum

Müssen wir uns noch die Situation für die Standorte angucken an denen wir feuchte Böden gemessen haben. Diese Standorte sind in Tabelle G.3 aufgeführt.

Die Mehrheitsklasse ist hier offensichtlich N, und die Fehlerrate liegt bei 1/5.

Auf den Trainingsfehler kommen wir, wenn wir die einzelnen Fehlerraten entsprechend gewichten und addieren:

$$\frac{11}{16} * \frac{4}{11} + \frac{5}{16} * \frac{1}{5} = \frac{5}{16}$$

**Betrachtung nach Säuregehalt:** Die Betrachtung des Säuregehaltes verläuft analog zu der der Bodenfeuchte. Einziger Unterschied ist hier, dass wir drei anstelle zwei diskrete Werte unterscheiden. Von daher kommen in der letzten Formel drei Summanden vor:

$$\frac{5}{16} * \frac{2}{5} + \frac{7}{16} * \frac{3}{7} + \frac{4}{16} * \frac{2}{4} = \frac{7}{16}$$

**Betrachtung nach Temperatur:** Hier tritt die Besonderheit auf, dass der Wertebereich nicht diskret ist und so unterteilen diesen Bereich in alle Temperaturen über 6.5 Grad und in alle die darunter liegen. Somit besteht unsere folgende Partition aus zwei Mengen:

$$\frac{5}{16} * \frac{2}{5} + \frac{11}{16} * \frac{5}{11} = \frac{7}{16}$$

Zum Abschluss soll noch über eine Möglichkeit geschrieben werden, die sogenannte Überanpassung zu verringern. Dazu kann man den Baum „stutzen“. Dabei wird die Tiefe des Baumes partiell verkürzt. Wieso stellt dies eine evtl. Verringerung einer möglichen

Überanpassung dar? Ein Stutzen des Baumes verkürzt Pfade und verkürzte Pfade machen Prognosen weniger differenziert. Damit kann eine zu genaue Anpassung des Baumes an die Trainingsdaten vermieden werden. Umgekehrt kann man aber auch durch zu starkes Stutzen den Wahren Fehler erhöhen. Welche Möglichkeiten haben wir um den wahren Fehler abschätzen zu können? Wir könnten ihn schätzen indem wir bei der Konstruktion des Baumes nur auf eine zufällig ausgewählte Teilmenge der Trainingsdaten zurückgreifen und den Rest zur Bestimmung des Wahren Fehlers nutzen (die sogenannte Validierungsmenge).

#### Literaturangaben zur Vertiefung

Görz, 2000 G. Görz, C.-R. Rollinger, J. Schneeberger. Handbuch der Künstlichen Intelligenz. Ab Seite 526.

Cleve, 2008 Jürgen Cleve, Vorlesungsskript „Wissensextraktion / Data Mining“ Abschnitt 6.3 (http://www.wi.hs-wismar.de/čleve/vorl/dm2160207807/skript/node1.html)

Murthy, 1998 Sreerama K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey

#### Instanzlernen

Dieses Verfahren kommt aus der Statistik und ist dort als das “Nächste Nachbarn”-Verfahren bekannt. Pionier auf diesem Gebit ist David Aha, 1991. Dieses Verfahren funktioniert auf sehr einfache Weise.

Es werden einfach alle Beispiele gespeichert. Wenn dann eine Prognose für eine unbekannte Eingabe abgegeben werden muss, das wird aus den gespeicherten Beispielen einfach das genommen was der Anfrage am nächsten kommt. Die diesem Beispiel zugehörende Klasse wird dann als Vorhersage genommen. Aus diesem Grund auch der Name „Nächste Nachbarn“ Verfahren. Hier fällt auch schon gleich grundsätzlicher Unterschied zum Entscheidungsbaumlernen auf. Während bei der Baummethode der Lernvorgang im Compilieren des Baumes besteht – also die Arbeit im Regelfall schon vor den ersten Anfragen passiert – passiert bis zu der ersten Anfrage beim instanzbasiertem Lernen nichts anderes als das Speichern der Trainingsdaten. Im Falle einer Anfrage wird nun versucht, das Ergebnis des Beispieles vorherzusagen, dass der Anfrage am nächsten liegt. Was „am nächsten“ bedeutet, ist mit die wichtigste Frage, an der sich die Güte der Vorhersage später zeigt.

#### Literaturangaben zur Vertiefung

Görz, 2000 G. Görz, C.-R. Rollinger, J. Schneeberger. Handbuch der Künstlichen Intelligenz. Ab Seite 533.

Cleve, 2008 Jürgen Cleve, Vorlesungsskript „Wissensextraktion / Data Mining“ Abschnitt 6.1 (http://www.wi.hs-wismar.de/čleve/vorl/dm2160207807/skript/node1.html)

Aha et al, 1991 David Aha, Dennis Kibler und Marc Albert. Instance-Based Learning Algorithms. Machine Learning, 6:37 - 66, 1991

### Assoziationsregeln

#### Einführung

Bei dem Finden von Assoziationsregeln handelt es sich um eine populäre Methode aus dem Bereich Data-Mining.

Es wird das Finden von Assoziationsregeln an folgendem Anwendungsfall erläutert: Gegeben ist eine Datenbank in der eine Menge von Einkaufsvorgängen (Transaktionen) gespeichert sind:

- {Grillfleisch, Einweggeschirr, Bier, Zahnpaste}
- {Grillfleisch, Mehrweggeschirr, Wein, Klopapier}
- {Grillfleisch, Zigaretten, Salami, Milch}
- {Grillfleisch, Einweggeschirr Bier, Fladenbrot}

Ziel des Assoziationsregelfindens ist nun, bestimmte Korrelationen zwischen einzelnen Artikel zu finden. Also Aussagen der Art: „Wer Grillfleisch und Einweggeschirr kauft, der ist wohl auch an Bier interessiert."

In unseren Beispieldaten trifft dies schon einmal auf zwei Einkaufvorgänge zu.

Damit sind wir soweit einen wichtige Anforderung – die an Assoziationsregeln gestellt wird – zu erläutern:

**Support oder auch Minimalhäufigkeit:** Eine übliche Größenordnung für die Minimalhäufigkeit, liegt bei 1% und würde in diesem Fall für jede Assoziationsregel fordern, dass mindestens 1% aller Transaktionen die in der Regel aufgeführten Artikel als Teilmenge haben. Für unsere Beispielregel heißt dies, dass mindestens 1% aller Einkaufsvorgänge Bier, Einweggeschirr und Grillfleisch beinhalten.

Diese Regel ist notwendig, damit eine Assoziationsregeln nicht auf einer zu kleinen Basis berechnet wird. Denn wenn nur ein Kunde z.B. Zigaretten und Kugelschreiber gekauft hat, dann wäre es sicher unvernünftig daraus eine allgemeine Regel abzuleiten.

Die zweite wichtige Anforderung an Assoziationsregeln, ist die der Konfidenz:

**Minimalkonfidenz:** Eine übliche Größenordnung ist z.B. 50%. Wir definieren den linken Teil der Assoziationsregel als Prämisse ( Grillfleisch und Einweggeschirr) und den rechte Teil als Konklusion. Eine Minimalkonfidenz von 50% würde fordern dass mindestens 50% aller Transaktionen die die Prämisse enthalten auch die Konklusion enthalten müssen. Für unsere Beispielregel heißt dies, dass mindestens 50% aller Einkaufsvorgänge die Grillfleisch und Einweggeschirr enthalten auch Bier enthalten müssen.

Diese Regel ist ebenfalls notwendig um die Relevanz einer Regel zu sichern. Denn wenn wie in unseren Beispieldaten jeder Grillfleisch gekauft hat, dann wäre es sicher falsch einen allgemeinen Zusammenhang von Grillfleisch und Milch zu konstruieren, da dies nur einmal zusammen gekauft wurde.

Wie kann das Finden von Assoziationsregeln formal definiert werden? Wir betrachten einen Definition aus [GR00, Görz, 2000]

> Sei I eine Menge von Objekten („items") und sei T eine Menge von Transaktionen, wobei $\forall t \in T : t \subseteq I$. Sei weiterhin $s_{min} \in [0, 1]$ eine benutzergegebene Minimalhäufigkeit („minimal support") und $c_{min} \in [0, 1]$ eine

benutzergegebene Minimalkonfidenz. Bei der Lernaufgabe Finden von Assoziationsregeln sind, gegeben $I, T, s_{min}$ und $c_{min}$, alle Regeln der folgenden Form zu finden:

$$X \rightarrow Y$$

wobei $X \subseteq IundY \subseteq IundX \cup Y = \{\}$ und es gilt

$$s(r) := \frac{|\{t \in TX \bigcup Y \in t\}|}{|T|} \geq s_{min}$$

$$c(r) := \frac{|\{t \in T|X \bigcup Y \in t\}|}{\{t \in T|X \in t\}|} \geq c_{min}$$

Im folgenden wird die Funktionsweise des Standardalgorithmus skizziert.

### Der Apriori-Algorithmus

Dieser Algorithmus findet alle Regeln wie sie im vorherigen Abschnitt definiert worden sind. Um effizient zu sein, nutzte er vor allem aus, dass der Raum aller Assoziationsregeln geordnet ist.

Dieser Ordnung zufolge kann man zum Beispiel schließen, dass alle Regeln die durch Verlängerung entstanden sind, seltener vorkommen als die entsprechend Kürzeren. Verlängerung bedeutet an dieser Stelle, dass man einer bestehenden gültigen Assoziationsregel in der Prämisse oder Konklussion ein Element anhängt. Also z.B. aus der Regel

Grillfleisch, Bier $\longrightarrow$ Fladenbrot

beispielsweise

Grillfleisch, Bier $\longrightarrow$ Fladenbrot, Kohle

machen würde.

Wenn eine Verlängerung eine der Anforderungen an unsere Assoziationsregeln verletzt, dann tun das auch alle weiteren Verlängerungen selbiger. Dadurch kann man mit den kürzest vorstellbaren Regeln anfangen und dies solange verlängern bis eine der Anforderungen verletzt wird.

Zum Schluss soll noch etwas zur Performanz des Apriorialgorithmus gesagt werden. Die Laufzeit hängt stark von der Struktur der Transaktionen ab. Genauer wie viele Teiltransaktionen auch wirklich häufig sind. Es ist nicht auszuschließen, dass exponentiell viele Teilmengen häufig sind. In diesem Fall ist die Laufzeit des Algorithmus natürlich auch exponentiell. Wenn wir uns aber unser Beispiel anschauen, bei dem wir Einkäufe als Transaktionen betrachten, so wird schnell klar, dass dies hier aller Wahrscheinlichkeit nach nicht der Fall sein wird, da ein einzelner Kunde immer nur einen sehr kleinen Anteil aller Artikel gleichzeitig kauft.

### Literaturangaben zur Vertiefung

Görz, 2000 G. Görz, C.-R. Rollinger, J. Schneeberger. Handbuch der Künstlichen Intelligenz. Ab Seite 568.

Cleve, 2008 Jürgen Cleve, Vorlesungsskript „Wissensextraktion / Data Mining“ Abschnitt 6.4 (http://www.wi.hs-wismar.de/ cleve/vorl/dm2160207807/skript/node1.html)

Agrawal et al., 1996 Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hanni Toivonen und A. Inkeri Verkamo. Fast Discovery of Association Rules. In Usama M. Fayyad, Gregory Piatetsky-Shapio, Padhraic Smyth und Ramasamy Uthurusamy (Hg.), Advances in Knowledge Discovery an Data Mining, Kapitel 12, Seite 307. AAAI/-MIT Press, Cambridge, USA, 1996.

### G.2.2 Unüberwachtes Lernen

#### Clustering

Bei der Clusteranalyse geht es darum, eine Menge von Eingabeinstanzen in Teilgruppen aufzuteilen. Dabei sollten alle Instanzen einer Gruppe (Cluster) sich möglichst ähnlich sein und alle Instanzen aus verschiedenen Klassen sollten sich möglichst unähnlich sein. Anwendung findet die Clusteranalyse z.B. auf Kundendatenbanken. Dort möchten Unternehmen ihre Kunden häufig in bestimmte Klassen einteilen, um sie gezielter mit Marketinginstrumenten anzusprechen.

**Definition Clusteranalyse:** [GR00, Görz, 2000. Seite 584]

Sei $X$ ein Instanzenraum und $S \subseteq X$ eine Menge von Instanzen. Sei weiterhin

$$dist : X \times X \rightarrow R^{+}$$

eine Abstandsfunktion, und

$$q : 2^{2^{X}} \rightarrow R$$

eine Qualitätsfunktion. Gegeben $S$, *dist* und $q$ besteht die Aufgabe der Clusteranalyse darin, eine Clusterung

$$C = \{C_1, ..., C_k\}, \text{ wobei } C_i \subseteq S \text{ fuer alle i} = 1, ..., k,$$

zu finden, so dass $q(C)$ maximiert wird.

Es gibt verschiedene Algorithmen der Clusteranalyse. Welche anwendbar sind, ist nicht zuletzt von der Beschaffenheit des Instanzraumes abhängig. Wenn es im Instanzraum möglich ist für jede beliebige Teilmenge von Instanzen deren Zentrum zu berechen, dann bietet sich z.B. das $k$-Means-Verfahren an. Als Zentrum gilt hier derjenige Punkt für den gilt, dass er von allen möglichen Punkten, den geringsten durchschnittlichen Abstand zu allen Instanzen der Gruppe hat. Bei allen rein numerischen Instanzräumen ist dies möglich.

In der Abbildung 54 ist der Fall dargestellt, dass die Eingabeinstanzen über zwei numerische Werte definiert sind. Die drei unausgefüllten Sterne, stellen die Eingabeinstanzen dar und der ausgefüllte Stern das Zentrum.

#### Grundidee des $k$-Means-Verfahren

- Es werden $k$ Instanzen $z_1 ... z_k$ zufällig aus unser Gesamtmenge an Intanzen ausgewählt.

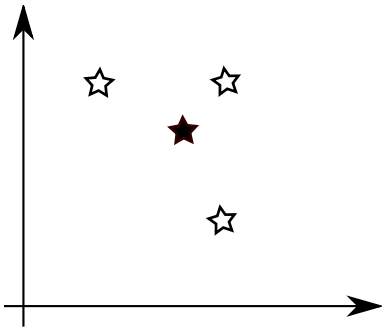


**Abbildung 54:** Zentrum von drei Instanzen

- Dann wird folgende Schleife solange wiederholt bis die Qualität des Clusters nicht mehr besser wird.
  - Füge jedem Cluster $C_i$ alle Punkte aus $S$ hinzu, für die der Abstand zum Zentrum am kleinsten ist.
  - Bestimme dann die Qualität des Clusters

Diese Verfahren terminiert sehr schnell. Es ist zu bemerken, dass das $k$-Means-Verfahren in lokalen Optima steckenbleiben bleiben kann. Wesentlich ist hier die zufällig gewählte Menge von Punkten am Anfang des Algorithmus. Hier setzen verschiedene Optimierungsversuche an.

#### Literaturangaben zur Vertiefung

Görz, 2000 G. Görz, C.-R. Rollinger, J. Schneeberger. Handbuch der Künstlichen Intelligenz. Ab Seite 583.

Cleve, 2008 Jürgen Cleve, Vorlesungsskript „Wissensextraktion / Data Mining“ Abschnitt 4.2 (http://www.wi.hs-wismar.de/ cleve/vorl/dm2160207807/skript/node1.html)

MacQueen, 1967 J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symp. on Mathematical Statistics and Probability, 1967

## G.3 Ideen zur Anwendungen in Spielen

### G.3.1 Entscheidungsbaumlernen bei PacMan

Ich habe das Baumlernen für das Aktionspiel aus folgenden Gründen ausgewählt. Zum einen hat es kurze Latenzzeiten (z.B. verglichen mit der Instanzmethode), da der Hauptzeitaufwand beim Kompilieren des Baumes anfällt. Zum anderen ist ja auch ein Thema dieser Seminararbeit bisher zu kurz gekommen. Nämlich das, der regelbasierten Steuerung. Gerade am Beispiel des Baumlernens kann man dies gut erläutern.

Eine Möglichkeit die KI zu verbessern, könnte zum Beispiel folgende sein:

Es werden verschiedene Werte – die man für relevant betrachtet – aus Sicht eines jeden Geistes protokolliert. Dies könnte zum Beispiel sein, ob er sich in der Nähe einer Pille befindet, ob der Gegner eine Pille gerade gegessen hat, wie viele andere Geister gerade in der Nähe sind usw.. Diese Werte, werden dann zum Beispiel jede Sekunde erfasst und gespeichert. Am Ende eines Levels erfolgt dann folgende Auswertung: Es wird – für alle

Situationen in denen ein Geist aufgefuttert wurde, oder den Gegner aufgefuttert hat - sagen wir mal 5 Sekunden in die Vergangenheit geschaut und der dort herrschende Zustand als Eingabetupel für unser Funktionslernen aus Beispielen genutzt. Wenn der Geist den Gegner gegessen hat, dann hat wird als Funktionsergebnis 1 angenommen und sonst 0. Daraus wird dann – wie eingangs beschrieben – der Baum kompiliert. In diesem Baum werden dann – von den Blättern aufwärts – alle Pfade gelöscht, die nur noch zu einem mit „0“ Blatt führen können.

So trainiert, kann dann die Geister-KI den Baum durchlaufen und basierend auf dem aktuellen Zustand, Entscheidungen treffen, die den Zustand dahingehend verändern, dass er eher in einer 1-Senke führt.

### G.3.2 Instanzlernen bei Schnick-Schnack-Schnuck

Beim Instanzlernen fällt bei den eigentlichen Klassifizierungsanfragen die eigentliche Arbeit an und von daher ist die Antwortzeit hier relativ hoch. Es bietet sich von daher auch eher für Spieltypen an, bei denen dies nicht so sehr ins Gewicht fällt. Zum Beispiel bei Schnick-Schnack-Schnuck. Auch wenn es in wesentlichen Teilen ein Glückspiel zu sein scheint, so haben viele menschliche Spieler eine Art „Strategie“ nach der sie die Reihenfolge – teils in Abhängigkeit vom Erfolg der vorherigen Wahl, teils ohne – bestimmen in der sie das nächste Bild präsentieren.

Wenn eine KI die Chance hat, häufig gegen ein und denselben Gegner zu spielen, könnte sie also sagen wir mal die jeweils letzten 10 Entscheidungen inklusive dem Erfolg/Misserfolg des Gegners speichern, dies als eine Eingabeinstanz für das Instanzlernen auffassen und dann immer zur Vorhersage des nächsten Zuges das Instanzlernverfahren benutzen.

# H Neuronale Netze

## H.1 Menschliches Gehirn

Eine genaue Vorstellung über den Aufbau und die Funktionsweise des menschlichen Gehirns, nämlich als ein komplexes Geflecht vernetzter Zellen, die untereinander Signale austauschen, wurde erst zwischen Ende des 19. und Anfang des 20. Jahrhunderts durch die Arbeiten von Camillo Golgi und Santiago Ramòn y Cajal bekannt, die 1906 dafür den Nobelpreis in Medizin erhielten. Nach heutigem Stand des Wissens werden Daten im Gehirn auf zwei funktionalen Ebenen im Gehirn repräsentiert, der sogenannten 'langsamen' bzw. 'schnellen' Ebene [RMS91]. Wie der Name andeutet, repräsentiert die schnelle Ebene Daten, die sich ständig verändern können. Sie wird durch die augenblicklichen Aktivitätsmuster der Neuronen gebildet. Diese Aktivitätsmuster enstehen einerseits durch die Beeinflussung durch Sinnesrezeptoren, andererseits durch gegenseitige 'Hemmung' oder 'Erregung' der Neuronen untereinander. Veränderungen in der schnellen Ebene, die für unmittelbare Wahrnehmungen und vermutlich für das Kurzzeitgedächtnis zuständig ist, spielen sich im Sekundenbereich ab. Bei der langsamen Ebene hingegen dauern Veränderungen Minuten bis Tage, wenn überhaupt Veränderungen stattfinden. Diese Ebene beeinhaltet insbesondere das Langzeitgedächtnis. Die Veränderungen beziehen sich hier hauptsächlich auf Veränderung der Synapsen, eine Eigenschaft, die erstmals 1949 von Donald Hebb entdeckt wurde, und auch 'synaptische Plastizität' genannt wird. Veränderungen auf den beiden Ebenen sind in beide Richtungen gekoppelt. So ändern einerseits die Neuronenaktivitäten die Verbindungen der Neuronen, andererseits wirken sich die Verbindungen der Neuronen auf die Neuronenaktivitäten aus. Vergleicht man die Reaktionsgeschwindigkeiten der Neuronen (quasi die 'Schaltelemente') im Gehirn, die bei ca. 1ms liegen, sowie die Signallaufzeiten von ca. 10 m/s an den Axonen, die man sich ungefähr als 'Ausgabeleitungen' der Neuronen vorstellen kann, mit denen aktueller Computer, sind diese doch deutlich unterlegen. Die Leistungsfähigkeit des menschlichen Gehirns muss also in massiver Parallelverarbeitung begründet sein. Weitere Besonderheiten des menschlichen Gehirns sind einerseits die große Fehlertoleranz gegenüber dem Ausfall einzelner Neuronen, sowie die Aufhebung der Trennung zwischen Daten und Algorithmen. Letzteres findet sich interessanterweise in der Programmiersprache LISP wieder, wo auf unabhängigem Wege ein ähnlicher Ansatz zustande kam.

## H.2 Biologischer Hintergrund

Sitz der intelligenten Leistungen beim Menschen ist die Hirnrinde, genauer gesagt der Neocortex, der 90% der Großhirnrinde ausmacht. Der Neocortex ist entwicklungsgeschichtlich der jüngste Teil des menschlichen Gehirns und ist nur bei Säugetieren vorhanden. Desweiteren existieren innerhalb mehrere verschiedene Felder mit Teilaufgaben, so z.B. der visuelle Cortex, der motorische Cortex und einige weitere. Unter jedem Quadratmillimeter der Großhirnrinde befinden sich ca. 100000 eng miteinander vernetzte Neuronen, die geschätzte Gesamtzahl der Neuronen im menschlichen Gehirn liegt zwischen $10^{11}$ und $10^{12}$, also zwischen 100 Milliarden und einer Billion. Ein Neuron, oder auf Deutsch auch Nervenzelle, setzt sich aus drei Hauptstrukturen zusammen: dem Dendriten, dem Zellkörper, auch Soma genannt, und dem Axon, welche grob den Funktionen Eingabe, Verarbeitung und Ausgabe entsprechen. Der Dendrit ist eine Baumartige Struktur, wo die Ausgangssignale der umgebenden bzw. verbundenen Neuronen in Form eines elektrischen Potentials summiert werden. Überschreitet das Potential einen

bestimmten Schwellenwert, so erzeugt der Zellkörper einen kurzen elektrischen Nadelimpuls (Spike), der vom Axon weitergeleitet wird. Das Axon ist eine Nervenfaser, die von wenigen Millimeterbruchteilen bis zu mehreren Metern lang sein kann. Auch sie verzweigt sich, und führt den Impuls somit bis zu mehreren tausend Zielneuronen zu. Hierfür setzt das Axon meistens an den Dendriten an, in manchen Fällen aber auch direkt am Soma des Zielneurons. Die entsprechende Kontaktstelle heißt Synapse. Die meisten Synapsen sind chemische Synapsen, d.h. sie wandeln ein elektrisches Signal in ein chemisches um, weswegen die Begriffe meist synonym verwendet werden. Darüber hinaus arbeiten die meisten Synapsen unidirektional vom prä- zum postsynaptischen Teil, also vom Axon in Richtung Dendrit/Soma. Die Wandlung des elektrischen in ein chemisches Signal erfolgt dadurch, dass die elektrische Entladung an der Synapse zur Ausschüttung von Neurotransmittern führt. Die Neurotransmitter wiederum werden an Rezeptoren gebunden, wodurch, vereinfacht gesagt, ein elektrisches Potential am Zielneuron hervorrgerufen wird, welches entweder eine hemmende (inhibitorische) oder erregende/aktivierende (exzitatorische) Wirkung haben kann. Obwohl es eine Vielzahl verschiedener Neuronen gibt, lassen sie sich in zwei große Hauptklassen unterteilen: die Pyramidenzellen und die Sternzellen. Die Pyramidenzellen, deren Gesamtanteil sich auf ca. 60% beläuft, haben meist Axonen mit großer Reichweite und exzitatorischen Synapsen, während die Sternzellen (ca. 40% aller Neuronen) nur in unmittelbarer Umgebung sternförmig verästelt sind und meist inhibitorische Wirkung haben. Nach heute weit verbreiteter Ansicht sind die Wesentlichen Informationen im Aktivitätszustand der Pyramidenzellen kodiert, und die Sternzellen dienen der Stabilisierung [RMS91]. Im Gehirn existieren oft Zusammenschlüsse von Neuronen zu Gruppen, die höhere funktionelle Einheiten bilden, die sogenannten 'Mikrosäulen', wobei ein einzelnes Neuron nicht fest *einer* Einheit zugeordnet sein muss, sondern ein gradueller Übergang vorliegen kann. Die Mikrosäulen werden wiederum zu größeren Einheiten zusammengeschlossen, den 'Rindenfeldern', die hochparallelisierte Spezialmodule für besondere Teilaufgaben darstellen. So gibt es z.B. im Sehcortex Rindenfelder für die Analyse von Kantenorientierungen, Farbtönen oder Geschwindigkeit. Man unterscheidet drei Hauptgruppen von Rindenfeldern, die aber jeweils einen ähnlichen Aufbau haben:

1. primäre und sekundäre Felder: die Eingabe kommt direkt von Sinnesrezeptoren oder primären Rindenfeldern
2. Assoziationsfelder: die Eingabe ist von prim. und sek. Rindenfeldern vorverarbeitet und stammt von verschiedenen Sinnesmodalitäten
3. primäre und sekundäre Motorfelder: steuern die Muskulatur (über nichtcortikale Zwischenstationen) bzw. primäre Motorfelder an

Die Verschaltung der Module im Gehirn erfolgt weitgehend topographisch, benachbarte Neuronen sind miteinander verbunden.

## H.3 Modelle

### H.3.1 McCulloch-Pitts-Zelle

Eines der ersten Modelle eines Neurons stammt aus dem Jahre 1943 und wurde von Warren McCulloch und Walter Pitts etwickelt, die vorschlugen, ein Neuron als ein logisches Schwellenwertelement darzustellen. Die McCulloch-Pitts-Zelle stellt das einfachste Neuronenmodell dar und verarbeitet nur Binärsignale. Es verfügt über n binäre Eingangsleitungen $x_i$, sowie eine Ausgangsleitung. Errengende und hemmende Synapsen werden modelliert durch ein Gewicht je Eingangsleitung $w_i \in \{-1, 1\}$, welches mit dieser multipliziert wird. Die Produkte werden aufsummiert und mit dem Schwellenwert

s der Zelle verglichen. Übersteigt die Summe den Schwellenwert, so ist das Neuron erregt, und an der Ausgabeleitung liegt der Wert 1 an. Andernfalls befindet sich das Neuron im Ruhezustand mit dem Wert 0 als Ausgabe. Mittels einer solchen Zelle ist es möglich, AND-, OR- und NOT-Gatter aufzubauen, so dass sich jede beliebige boolsche Funktion darstellen lässt. Zwei wesentliche Probleme dieses Modells waren jedoch einerseits die mangelnde Fehlertoleranz gegenüber dem Ausfall einzelner Neuronen, die das menschliche Gehirn auszeichnet, d.h. ein Netzwerk aus McCulloch-Pitts-Zellen ist auf die fehlerfreie Funktion aller Elemente angewiesen. Andererseits war unklar, wie die Verschaltung der einzelnen Zellen zustande kommen sollte, besonders im Hinblick auf den Aspekt des Lernens. Ein Lösungsansatz hierfür erbrachte Hebb 1949 in seinem Buch „The Organization of Behaviour“ mit der bereits erwähnten „synaptischen Plastizität“: Die Verschaltung zweier Neuronen durch eine Synapse verstärkt sich mit steigender gemeinsamer Aktivität durch Anpassung der Synapse. Dies ist auch als *Hebbsche Lernregel* bekannt. Mathematisch formuliert lautet sie

$$\Delta w_i = \epsilon \cdot y(\vec{x}) \cdot x_i$$

für die Änderung der Synapsenstärken eines Neurons. Dabei bezeichnet $\vec{x}$ den Eingabevektor, $y(\vec{x})$ die Erregung des Neurons und die Konstante $\epsilon > 0$ bemisst die Größe eines einzelnen Lernschrittes.

### H.3.2 Perzeptron

Ein wichtiger Schritt für das Rechnerbasierte Lernen gelang Frank Rosenblatt im Jahre 1958 mit dem Konzept des Perzeptrons. Es besteht aus einer festen Anzahl von $n$ Elementen und $l$ Eingabeleitungen. Letztere bilden den Eingansvektor der Länge $l$, welchen jedes Element als Eingabe erhält. Die einzelnen Elemente entsprechen von ihrer Funktion her einer McCulloch-Pitts-Zelle, es gibt jedoch kleine Unterschiede. So enthält ein Element keinen expliziten Schwellenwert $s$, dieser ist implizit eingebaut, nämlich als eine zusätzliche Eingabeleitung mit dem konstanten Wert 1 mit einem zugehörigen Gewicht -s.
Anstelle von

$$w_1 \cdot x_1 + \cdot s + w_n \cdot x_n \geq s$$

wird also

$$w_1 \cdot x_1 + \cdots + w_n \cdot x_n - s \geq 0$$

bzw.

$$w_1 \cdot x_1 + \cdots + w_n \cdot x_n + w^* \cdot x^* \geq 0 \text{ mit } w^* = -s \; und \; x^* = 1$$

gerechnet.
Die Gewichte $w$ sind reelle Zahlen, die Ausgabe der einzelnen Elemente ist 0 oder 1. Die Eingabevektoren bilden Eingabemuster, von denen jedes Muster einer von $n$ Musterklassen angehört. Das Training des Perzeptrons erfolgt durch Klassifikationsbeispiele mit jeweiliger Angabe der zugehörigen Klasse. Während des Trainings werden die Gewichte so angepasst, dass jedes Element nur noch auf das Eingabemuster „seiner“ Klasse reagiert. Gibt ein Element eine 0 aus, obwohl eine 1 richtig wäre, so werden die Gewichte inkrementiert. Der umgekehrte Fall verläuft analog. Ein Problem ist hierbei, das die Existenz einer Lösung nicht gesichert ist, also ob es derartige Gewichte, die die Klassifikation lösen, überhaupt gibt. Dies führt zum Problem der linearen Separierbarkeit, bezüglich des Perzeptron auch XOR-Problem genannt. So ist ein Neuron des Perzeptrons mit zwei Eingängen nicht in der Lage, den XOR-Operator darzustellen. Bei binären Funktionen bedeutet lineare Separierbarkeit, dass die Urbilder von 0 und 1 separierbar sind. So ist die OR-Funktion linear separierbar, die XOR-Funktion jedoch nicht.

| | OR | XOR |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 1 | 1 |
| 10 | 1 | 1 |
| 11 | 1 | 0 |

**Abbildung 55:** Die OR- und XOR-Funktion. OR ist linear separierbar, XOR nicht

### H.3.3 Assoziative Speicher

Auch im Vergleich zum menschlichen Gehirn vermochte das Perzeptron eine wichtige Eigenschaft nicht zu leisten, nämlich die Fehlertoleranz gegenüber Ausfällen einzelner Elemente. Um dies zu gewährleisten, müssten die Informationen auf meheren Elementen verteilt gespeichert werden, so dass der Ausfall eines einzelnen Elementes lediglich eine Verschlechterung, nicht jedoch einen kompletten Verlust der Information zur Folge hätte. Ein Lösungsansatz für dieses Problem kam 1969 von Willshaw, Bunemann und Longuet-Higgins. In ihrem Modell werden Informationen in Form eines Musterpaars $(x, y)$ aus binären Vektoren $x \in \mathbb{B}^L$ und $y \in \mathbb{B}^N$ dargestellt, wobei $x$ das Eingabe- und $y$ das Ausgabemuster ist, $x$ fungiert also als 'Abrufschlüssel' für das Muster $y$. Die Speicherung erfolgt in einer $L \times N$ Matrix der Synapsenstärken, wobei $L$ die Dimension der Eingabemuster $x$ und $N$ die der Ausgabemuster $y$ ist. $L$ sollte dabei groß sein, während $N$ auch klein sein darf. Letzteres bestimmt auch die Anzahl der benötigten Schwellenwertelemente, die mit denen der McCulloch-Pitts-Zelle identisch sind, also die Funktion

$$\sum_{i=1}^{L} w_{ri} x_i - s_r > 0,\ i \in \{1 \dots L\},\ r \in \{1 \dots N\}$$

zur Bestimmung der $y_r$ berechnen. $s_r$ bezeichnet die Schwellenwerte. Initial haben alle Elemente $w_{ri}$ der Matrix den Wert 0, bei der Speicherung eines Musters werden alle $w_{ri}$, für die $x_i = 1$ und $y_r = 1$ ist, auf 1 gesetzt. Dadurch ist es natürlich möglich, dass mehrere Muster sich überlagern und der Abruf eines Musters mit einem Schlüssel $x$ zu einer fehlerhaften 1 in selbigem führt. Es kann jedoch gezeigt werden, dass dies bei einer hinreichend geringer Anzahl Einsen in den Eingaben nur mit kleiner Wahrscheinlichkeit vorkommt [RMS91]. Entscheidend für die Fehlertoleranz allerdings ist die Wahl der Schwellenwerte $s_r$. Wählt man einen hohen Schwellenwert wie z.B. $s_r = k - \frac{1}{2}$, wobei $k$ für die Anzahl der Einsen in den Eingaben stehen, so führt bereits das Fehlen einer Eins in der Eingabe oder der Ausfall einer einzigen Synapse zu schweren Fehlern im Ergebnis. Ein Herabsetzen des Schwellenwertes führt hingegen zur erwünschten Ausfalltoleranz, aber gleichzeitig zu einer höheren Fehlerwahrscheinlichkeit bzgl. der Assoziationen. Dies kann durch herabsetzen der Anzahl gespeicherter Muster kompensiert werden, was einer gerigeren Ausnutzung der Speicherkapazität entspricht.

### H.3.4 Hopfield Modell

Das Hopfield Modell stellt ein rückgekoppeltes Netz dar, d.h. die Ausgabeleitungen werden auf die Eingabeleitungen zurückgekoppelt, und das Ausgabemuster kann zum Eingabemuster beitragen. Ein besonderer Fall der Rückkopplung ist die sog. 'Autoassoziation' mit y=x, so dass jedes Eingabemuster sich selbst wieder als Ausgabe zugeordnet ist. Mit einem solchen Netz lassen sich auch unvollständige Muster wieder rekonstruieren. Wird also in ein Netz, in dem Musterpaare gespeichert sind, ein fehlerhaftes bzw. unvollständiges Muster eingegeben, so ist das Netz durch wiederholte Rückkopplung des jeweils leicht verbesserten Ausgabesignals in der Lage, dieses Muster sukzessiv zu verbessern und schließlich wieder vollständig herzustellen. Dies ist eine Eigenschaft,

die auch Merkmal biologischer Nervensysteme ist, die dafür optimiert sind, unvollständige Informationen zu bearbeiten und zu ergänzen. In seiner ursprünglichen Fassung benutzte das Hopfield Modell McCulloch-Pitts-Neuronen, in neueren Fassungen ist die Ausgabemenge der Neuronen jedoch zu $\{-1, 1\}$ statt $\{0, 1\}$ abgeändert worden [Hop82]. Jedes Neuron ist mit jedem anderen Neuron verbunden, so dass die Aktivität eines Neurons aus den Ausgaben aller anderer gebildet wird. Die Synapsenstärken sind hierbei symmetrisch, d.h. für zwei Neuronen $i$ und $j$ gilt $w_{ij} = w_{ji}$, diese Forderung garantiert die Lösbarkeit des Modells. Es stellt auch eine Analogie zu „wechselwirkenden Vielteilchensystemen der Statistischen Physik“ dar, was das Thema Neuronale Netze auch für diesen Fachbereich zugänglich und interessant machte, und umgekehrt Methoden daraus für Neuronale Netze nutzbar machte [Hop82]. Gravierende Nachteile des Modells sind einerseits die schlechte Eignung zur Speicherung ähnlicher oder korrelierter Muster, sowie die mangelhafte Invarianz gegenüber Transformationen der Muster. Die Beurteilung der Muster erfolgt über übereinstimmende Pixel, schon eine einfache Transformation, wie z.B. eine Verschiebung, können die Wiedererkennung unmöglich machen.

### H.3.5 Feed-Fordward-Netze, Backpropagation Algorithmus

Im Hopfield-Modell ist jedes Neuron mit jedem anderen verbunden, es existiert also keine innere Struktur. In Neuronalen Netzen ist jedoch üblicherweise eine Struktur vorhanden, oft z.B. ist eine Hintereinanderschaltung mehrerer Neuronenschichten zu finden. Eine solche Eigenschaft bietet ein vorwärtsgekoppeltes Netzwerk (feed-forward network), das prinzipiell einem Perzeptron mit mehreren Schichten entspricht und dadurch auch dessen Einschränkungen aufhebt. Es besteht aus einer Eingabeschicht, einer oder mehrerer innerer Schichten sowie einer Ausgabeschicht. Die Neuronen der innneren Schicht sind ‘isoliert‘, d.h. sie sind nicht mit den Ein- und Ausgabeleitungen verbunden. Der Aktivitätszustand dieser Neuronen ist also durch die innere Verschaltung und nicht durch die ‘Außenwelt’ bestimmt, jede Schicht erzeugt eine Transformation des Aktivitätsmusters der Vorgängerschicht. Die Eingänge der Neuronen der Eingangsschicht sind im Übrigen nicht gewichtet, sondern übernehmen die Eingangswerte direkt. Ein weiterer wesentlicher Unterschied von Feed-Forward-Netzen zum Perzeptron ist, dass die enthaltenen Neuronen üblicherweise keine binäre, sondern eine kontinuierliche Ausgabefunktion besitzen. Die Funktion ist eine sigmoide Funktion $\sigma(x)$, sie ist nichtnegativ, überall monoton steigend, und kovergiert für $x \to \pm\infty$ gegen 1 bzw. 0. Eine häufige Wahl für eine solche Funktion ist die Fermifunktion:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Wir gehen im Folgenden von einem dreilagigen Netz aus, bei dem die Indizes $i, j, k$ jeweils für Variablen der Eingangs-, der inneren bzw. der Ausgabeschicht stehen.

Die Ausgaben der Neuronen der Ausgabeschicht berechnen sich nach der Formel

$$s_k = \sigma(\sum_j w_{jk} s_j)$$

Gesucht sind nun Werte für die Synapsenstärken $w_{ij}$ und $w_{jk}$ derart, dass die Eingabevektoren korrekt auf die Ausgabevektoren abgebildet werden. Ein Ansatz zur Lösung des Problems, wie die einzelnen Gewichte der Synapsen zu bestimmen sind, ist der Backpropagation Algorithmus. Er geht zurück auf die Arbeiten von Werbos im Jahre 1974 [Wer74] bzw. von Rumelhart und Williams (1986) [RHW86] und gehört zur Kategorie der überwachten Lernverfahren. Die Bewertung der Güte der Abbildung geschieht hierbei durch die quadratische Fehlerfunktion E:

$$E = \frac{1}{2} \sum_k (y_k - s_k)^2$$

Hierbei bezeichnet $y$ die gewünschte Ausgabe und $s$ die tatsächliche Ausgabe des Netzes. Die Formel bezieht sich auf den Fall, dass nur mit einem Eingabevektor trainiert wird, für mehrere Vektoren müssten diese ebenfalls in der Formel aufsummiert werden. Die Konstante $\frac{1}{2}$ hat keinen Einfluss auf das Ergebnis, sie vereinfacht lediglich die Rechnung an späterer Stelle. Die Suche geeigneter Synapsenstärken entspricht der Minimierung der Funktion E. Der Backpropagation Algorithmus führt näherungsweise einen Gradientenabstieg für E durch. Die verwendete Lernregel entspricht dem Verfahren der sog. *Delta-Lernregel*, ist aber modifiziert im Vergleich zu dieser, da die Delta-Lernregel nur für einschichtige Netze geeigenet ist (wie auch die Hebbsche Lernregel). Sie betrachtet die Abweichung zwischen Soll- und Istwert:

$$\Delta w_{ab} = -\alpha \cdot \frac{\partial E}{\partial w_{ab}}$$

Der Backpropagation-Algorithmus läuft nun in den folgenden 3 Phasen ab:

1. Forward Pass
   Im Forward-Pass wird ein Eingabevektor (oder auch mehrere), auf den das Netz trainiert werden soll, in das Netz eingeben, zu dem dieses einen Ausgabevektor berechnet.

2. Bestimmung des Fehlers
   Da zu jeder Eingabe auch die gewünschte Ausgabe vorhanden ist, kann nun der Fehler mit Hilfe der Fehlerfunktion bestimmt werden, indem die Ausgabe des Netzes mit der richtigen Ausgabe verglichen wird. Liegt der Fehler oberhalb einer vorher gewählten Güteschwelle, wird das Netz im Backward Pass angepasst. Andernfalls wird die Trainingsphase an dieser Stelle abgebrochen.

3. Backward Pass
   Im dritten Schritt werden schließlich die Synapsengewichte angepasst. Die Änderung für Synapsen zwischen innerer Schicht und Ausgabeschicht berechnet sich durch
   $$\Delta w_{jk} = \alpha \cdot \epsilon_k \cdot s_j \cdot s_k(1 - s_k).$$
   Die tatsächliche Anpassung der Synapsengewichte geschieht nach der Formel
   $$w_{jk}^{neu} = w_{jk}^{alt} + \Delta w_{jk}.$$
   Die Änderung für Synapsen zwischen Eingabeschicht und innerer Schicht berechnet sich durch
   $$\Delta w_{ij} = \alpha \cdot \sum_k \epsilon_k \cdot s_i \cdot s_k(1 - s_k) \cdot w_{jk} \cdot s_j(1 - s_j),$$
   die Anpassung erfolgt analog. $\epsilon$ bezeichnet den $i$-ten Ausgabefehler:
   $$\epsilon_i = y_i - s_i(\vec{x}),$$
   und für die Ableitung der Fermifunktion, durch welche die Ausgabewerte $s_i, s_j, s_k$ berechnet werden, wurde folgende Substitution gemacht:
   $$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

Ein Problem, das bei diesem Verfahren auftreten kann, ist das der lokalen Minima. Je nach Wahl der initialen Synapsenstärken führt der Gradientenabstieg zum nächsten Minimum, und der Algorithmus bleibt stecken. Handelt es sich hier um ein lokales und kein absolutes Minimum, beendet der Backpropagation Algorithmus, ohne dass das berechnete Netzwerk seine Aufgabe richtig löst. Nichtsdestrotz stellt der Algorithmus aber einen bedeutenden Fortschritt im Vergleich zu früheren Ansätzen, wie z.B. dem Perzeptron, dar.

### H.3.6 Selbstorganisierende Karten

Bei allen bisherigen Modellen lag der Fokus auf den Verbindungen der Neuronen. Eine erste Form der Strukturierung lag mit der Gliederung in verschiedene Schichten bei den Feed-Forward Netzen vor, die Lage eines Neurons *innerhalb* einer Schicht war aber noch immer unbedeutend. Dies ändert sich mit den selbstorganisierenden Karten (self-organizing maps, SOM), bei denen die Anordnung der Neuronen eine wichtige Rolle spielt. Ziel dieses Modells ist eine Signalähnlichkeit der Neuronen in Lagenachbarschaft. Bei natürlichen Neuronalen Netzen können die Neuronen so über kurze Verbindungswege schnell kommunizieren, was ein wichtiger Faktor für die effiziente Parallelverarbeitung ist. Auf diesem Wege entstehen topographische Karten der Eingabesignale, die Ähnlichkeiten der Signale durch Lagerelationen der aktivierten Neuronen repräsentieren, was wiederum der Bildung interner Datenrepräsentationen entspricht. Derartige Ansätze hatten den Fokus allerdings eher auf biologischer Detailtreue als auf praktischer Anwendbarkeit. Erst das Modell von Kohonen lieferte einen abstrakteren und allgemeineren Ansatz zur Modellierung selbstorganisierender sensorischer Karten.

### H.3.7 Modell von Kohonen

Das Modell von Kohonen stellt eine topographische Merkmalskarte dar, d.h. die Reaktion auf einen bestimmten Reiz beschränkt sich auf Neuronen innerhalb eines gewissen räumlich lokalisierten Gebiets auf der Karte der Neuronen. Bei den Reizen kann es sich um ein einfaches Merkmal, wie etwa die Position auf einer (Sinnes-)Oberfläche (z.B. der Haut), aber auch um abstraktere Merkmale wie der Frequenz eines Tons oder ähnliches handeln. Das Modell von Kohonen benutzt in der Regel eine zweidimesionale Neuronenschicht $A$. Durch diese Schicht verlaufen $d$ Eingangfasern (Axone), die die Neuronen wie gehabt über synaptische Verbindungen ansteuern. Die Neuronen sind üblicherweise in einem regelmäßigen Gitter angeordnet und werden über ihren Ortsvektor $\vec{r} \in A$ identifiziert. Wie auch in den bisher betrachteten Modellen berechenen die Neuronen an ihren Dendriten die gewichtete Summe $\sum_l w_{rl} v_l$ der Eingangsfasern $v_l$, $l \in \{1 \dots d\}$ (also über alle Eingangsfasern). Hemmende Synapsen haben wiederum negative Werte, erregende positive. Die Aktivität eines Neurons wird durch seine mittlere *Spikefrequenz* $f_r^0$ beschrieben:

$$f_r^0(v) = \sigma\left(\sum_l w_{rl} v_l - \theta\right)$$

$\sigma(x)$ bezeichnet wieder eine sigmoide Funktion, $\theta$ stellt einen Schwellenwert dar, unterhalb dessen das Neuron nur noch schwach reagiert. Neben der Kopplung der Neuronen mit den Eingangsfasern existiert in Kohonens Modell noch eine Kopplung der Neuronen untereinander, wodurch eine Rückkopplung der Schicht auf sich selbst erreicht wird. Bezeichnet $g_{rr'}$ die Kopplungstärke von Neuron $r'$ zu Neuron $r$, so fließt der Wert $g_{rr'} f_{r'}$ in die Erregung von Neron $r$ ein. Unter Berücksichtigung aller verbundenen Neuronen entspricht die Erregung eines Neurons $r$ also der Lösung des nichtlinearen Gleichungssystems

$$f_r = \sigma\left(\sum_l w_{rl} v_l + \sum_{r'} g_{rr'} f_{r'} - \theta\right)$$

Die Rückkopplung ist häufig eine sog. *Umfeldhemmung*, d.h. sie ist für kurze Distanzen erregend (also $g_{rr'} > 0$) und für große Distanzen hemmend (also $g_{rr'} < 0$) Für die Anpassung der Synapsenstärken wird im Modell von Kohonen nicht die Wechselwirkung der kompletten Karte berechnet, sondern es geht vom Zentrum des Erregungsmaximums $r'$ ein Erregungssignal $h_{rr'}$ aus. Die Änderung der Synapsenstärken berechnet sich durch:

$$\delta w_{rl} = \epsilon(h_{rr'} v_l - h_{rr'} w_{rl})$$

Dies entspricht im Prinzip der Hebbschen Lernregel, ist jedoch ergänzt um einen Abklingterm. $\epsilon$ ist die Größe eines einzelnen Adaptionsschritts und liegt zwischen 0 und 1. Es wird oft als von der Zeit abhängige Funktion $\epsilon(t)$ gewählt, deren Wert mit der Zeit sinkt, um schnell auf richtige Synapsenwerte zu kommen. Da die genaue Form der Erregungsantwort $h_{rr'}$ keine entscheidende Rolle spielt und die hierfür nötige Auswertung des nichtlinearen Gleichungssystems aufwändig ist, wird sie in Abhängigkeit der Distanz $(r - r')$ mittels der Gaußglocke angenähert:

$$h_{rr'} = \exp\left(\frac{-(r-r')^2}{2\tau^2}\right)$$

$\tau$ ist der Radius der Funktion, in dem sich die Änderungen auf der Karte auswirken. Oft wird der Radius als eine von der Anzahl $t$ der Lernschritte abhängige Funktion $\tau(t)$ gewählt, die mit wachsendem t einen geringeren Wert zurückliefert. Der genaue Ablauf wird nun durch folgenden Algorithmus beschrieben:

0. **„Initialisierung“**
   Zunächst werden Startwerte für die Synapsenstärken $w_{rl}$ gewählt. Sofern sich nicht aus irgendwelchen Gründen bestimmte Werte anbieten, werden diese einfach zufällig gewählt.

1. **„Stimuluswahl“**
   Die in jedem Schritt anliegenden Eingangsreize werden als voneinander unabhängige Zufallvariable betrachtet. Entsprechend der zugehörigen Verteilungsdichte $P(\vec{v})$ wird nun ein zufälliger Vektor $\vec{v}$ gewählt, der das Eingabesignal repräsentiert.

2. **„Response“**
   In diesem Schritt wird nun das ‘Erregungszentrum’ $r'$ gesucht, das die Bedingung

   $$\|\vec{v} - \vec{w}_{r'}\| \leq \|\vec{v} - \vec{w}_r\| \qquad \text{für alle } \vec{r} \in A$$

   Wobei $\|\vec{x}\|$ für die Länge des Vektors $\vec{x}$ steht.

3. **„Adaptionsschritt“**
   Zuletzt wird der Lernschritt in Form einer Änderung der Synapsenstärken gemäß

   $$\vec{w}_r^{neu} = \vec{w}_r^{alt} + \epsilon h_{rr'}(\vec{v} - \vec{w}_r^{alt})$$

   durchgeführt. Schließlich fängt bei Schritt 1. die nächste Iteration an.

## H.4 Einsatz in Computerspielen

### H.4.1 Joebot

Ein Beispiel für den Einsatz von Neuronalen Netzen in Computerspielen ist der ‘Joebot’, ein Bot für den Strategie-Shooter ‘Counterstrike’, der von Johannes Lampel programmiert wurde. Es kommen zwei Neuronale Netze zum Einsatz, eins für den Kampf und eins für die Kollisionsdetektion bzw. -vermeidung.

Das Kampf-NN ist ein 6-6-6-5 Netz, es bekommt also 6 Eingaben und liefert dazu 5 Ausgaben. Die Eingaben sind: eigene Gesundheit, Entfernung des Gegners, Waffe des Gegners (beide nur, wenn der Gegner sichtbar ist), eigene Waffe, eigene Munition und die momentane Situation. Alle Eingabewerte liegen im Bereich $[-1, 1]$, Gesundheit und Munition sind in dieses Intevall abgebildete prozentuale Werte, während es für die Waffen festgelegte Werte gibt. So hat die schlechteste Waffe, das Messer, den Wert -1,

das Scharfschützengewehr, welches die beste Waffe darstellt, den Wert 1. Die momentane Situation berechnet sich aus den Parametern 'Anzahl der Gegner', 'Anzahl der Mitkämpfer' und der allgemeinen eigenen Einstellung des Bots, die entweder 'defensiv' oder 'aggressiv' sein kann. Die hieraus berechneten Ausgaben sind: Springen, Ducken, Verstecken, Links/Rechts und Laufen/Gehen. Für boolesche Ausgabewerte existiert hier jeweils eine Schwelle, so wird 'Springen' beispielsweise ab dem Wert 0.5 auf *true* gesetzt.

Das Neuronale Netz für die Kollision ist ein 3-3-1 Feed-Forward Netz. Die Eingabe kommt von drei 'Fühlern' des Bots, von denen einer gerade nach vorne 'zeigt', die anderen beiden nach links und rechts, jeweils in einem Winkel $\alpha$, der auf 35° gesetzt ist. Das Netz wird nur aktiv, wenn einer der drei Fühler ein Signal gibt, also wenn er auf ein Hindernis wie ein Objekt oder eine Wand trifft. Die Ausgabe des Netzes wird auf die Entscheidungen 'links', 'rechts' oder 'geradeaus' projiziert.

Das Trainieren des Kampf-Netzes schließlich geschah offline mittels vogegebener Muster. Um die Funktion des Netzes zu testen, sammelte der Autor alle Eingabemuster für das Netz während eines Spiels. Diese wurden in eine selbstorganisierende Karte eingegeben, welche selbige dann kategorisierte. Auf die so kategorisierte Karte wurden nun die benutzten Muster projeziert. Somit konnten die Verteilung der Gewichte und die Distanzen der Trainingseingaben und der Eingaben im Spiel erkannt werden. Große Distanzen der Muster deuteten beispielsweise auf 'vernachlässigte' Muster hin, und der Bot konnte dahingehend optimiert werden.

### H.4.2 NERO

Ein weiteres Beispiel ist NERO (Neuro Evolving Robotic Operatives), ein Spiel bzw. eine Plattform, entwickelt von der Neural Networks Research Group des Fachbereichs Informatik der University of Austin, Texas. NERO wurde mit dem Ziel entwickelt, die Mächtigkeit moderner maschineller Lerntechnologie zu demonstrieren. Außerdem wollten die Forscher eine Entwicklungs- und Benchmark-Plattform zur Verfügung stellen, die von anderen KI-Forschern benutzt werden kann. Das Projekt begann im Oktober 2003, die erste Version wurde im Juni 2005 veröffentlicht. Das Spiel an sich ähnelt bekannten Echtzeitstrategiespielen, weist jedoch einige entscheidende Unterschiede auf, weswegen die Entwickler es als ein eigenes, neues Genre ansehen, das sie „machine learning games" nennen. Der Spielverlauf teilt sich in zwei Phasen auf. Die erste Phase besteht aus dem taktischen Training der Roboter (Agenten) durch den Spieler. Während des Trainings kann der Spieler beispielsweise Objekte wie feindliche Geschütze oder Hindernisse wie Mauern aufstellen und so das Verhalten der Agenten für die Kampfphase darauf anpassen. Ist diese Phase abgeschlossen, treten die Roboter in der zweiten Phase gegen ein gegnerisches Team aus trainierten Robotern an. Dies kann etweder ein Computergegner oder auch ein von einem anderen Spieler trainiertes Team sein.

Jeder Roboter im Spiel hat ein eigenes Neuronales Netz, das sein „Gehirn" darstellt. Die Auswertung erfolgt durch einen Algorithmus namens „NEAT" (Neuro-Evolution of Augmenting Topologies). Es handelt sich hierbei um einen genetischen Algorithmus, es kommt also die Methode des bestärkenden Lernens (reinforcement learning) zum Einsatz, bei der gute Agenten belohnt und schlechte bestraft werden. Diese Bewertung geschieht durch den Spieler durch die sogenannte 'scoring control'. In NERO kommt eine Echtzeitvariante namens „rtNEAT" zum Einsatz. Diese ermöglicht eine kontinuierliche Auswertung einer kleinen Population (ca. 30 Agenten). Statt mit Generationen wird hierbei mit einer künstlichen Lebenszeit der Agenten gearbeitet. Das 'Gehirn' des schlechtesten Agenten der Population wird ersetzt durch die Kreuzung zweier guter „Gehirne" anderer Agenten, gute Gehirne werden für den späteren Gebrauch gespeichert.

# I Soziologie/Psychologie: Modelle für Emotionen/Verhalten von Spielern

## I.1 Einleitung

Spiele lösen Reize beim Benutzer aus, die dessen Spiel-Verhalten und -Motivation beeinflussen. Dabei kann ein positiver Affekt zu einer Weiterbeschäftigung, genauso wie ein negativer Affekt, wie etwa Frustration oder Ärger, zum Abbruch des Spieles führen.

Dabei unterscheiden sich Spiele deutlich von anderen Medien, da die Selbstwirksamkeit[1] einen zentralen Anreiz darstellt. Hierbei werden laut Lerntheorie Emotionen „durch das Vorhandensein oder Ausbleiben von Belohnungen und Strafen angeregt, intensiviert, herabgesetzt und auch beendet“[Mue07].

Hat der Spieler Erfolg durch sein Handeln, erhält er Belohnungen, durch Punkte oder Erreichen eines nächsthöheren Levels. Dies zieht Vergnügen und Hochgefühl nach sich und wird meistens durch grafische Effekte oder bekräftigende Soundeffekte intensiviert. Im Gegensatz dazu kann ein Ausbleiben oder Abzug von Belohnungen (z.B. Schwächung der Spielfigur) zu Frustration, Ärger oder Wut führen.

Hierbei bewegt sich (laut Fritz [FF03]) die Tätigkeit entlang einer 'Flow- vs. Frust-Spirale'. Dies bedeutet, dass ein Flow-Zustand, aus angenehmen Gefühlzustand heraus, zum weiterspielen animiert und der Frust-Zustand einen Antrieb des Wunsches nach Erfolg darstellt, solange die Frustration nicht zu groß wird und den Spielwunsch terminiert. Hierzu sollen die Begriffe Emotion und Verhalten genauer erläutert werden.

### Definition Verhalten [Lex07]

„Verhalten, im weiteren Sinn die Gesamtheit aller beobachtbaren (feststellbaren oder messbaren) Reaktionsweisen oder Zustandsänderungen von Materie, insbesondere das Reagieren lebender Strukturen auf Reize; im engeren Sinn die Gesamtheit aller Körperbewegungen, Körperhaltungen und des Ausdrucksverhaltens eines lebenden tierischen Organismus in seiner Umwelt.“

Aufgrund der Menge an Definitionen für Emotion, soll hier auf eine erschöpfende Übersicht verzichtet und nur eine Arbeitsdefinition gegeben werden.

### Arbeitsdefinition Emotion [Fun07]

**Arbeitsdefinition Teil 1: Beispiele**

„Emotionen sind Vorkommnisse von z.B. Freude, Traurigkeit, Ärger, Angst, Mitleid, Enttäuschung, Erleichterung, Stolz, Scham, Schuld, Neid sowie von weiteren Arten von Zuständen, die den genannten genügend ähnlich sind“

**Arbeitsdefinition Teil 2: Merkmale**

„

(a) **Emotionen sind aktuelle Zustände von Personen**

[1] Selbstwirksamkeit: aufgrund eigener Kompetenzen eine Handlung ausführen

Abgrenzung aktueller emotionaler Episoden (=Emotion; z.B. Angst) von emotionalen Dispositionen (z.B. Ängstlichkeit)

(b) **Emotionen unterscheiden sich nach Qualität und Intensität**
Art der Emotion (Freude, Wut) versus starke und schwache Ausprägung einer bereits bestimmten Qualität

(c) **Emotionen sind in der Regel objektgerichtet**
man freut sich über etwas, ist stolz auf etwas, etc.
das Ëtwasmmuß nicht konkret existieren, sondern kann auch vorgestellt sein (z.B. zukünftiges Ereignis "Prüfung")

(d) **Personen in einem emotionalen Zustand**
haben normalerweise ein charakteristisches Erleben, häufig treten auch bestimmte physiologische Veränderungen und Verhaltensweisen auf"

Somit sind Emotionen also Grundphänomene des individuellen, subjektiven Erlebens und stellen einen komplexen Befindlichkeitszustand des Menschen dar. Sie sind primär biologische Reaktionen, welche Bewertungen und Handlungs- bzw Verhaltensmöglichkeiten in Bezug von Erfahrung und Wünschen zur aktuellen Situation darstellen.

Auf Grund der Komplexheit von Emotionen soll der Tabelle „Abbildung I.1" noch eine Anregung für Eigenheiten und Dimensionen von Emotionen gegeben werden (inklusive einiger Beispiele in Klammern). Die Eigenschaften,in ihr ihrer Dimension, haben eine Ausprägung in zwei Richtungen. Als Beispiel wäre z.B. die Dimension „Richtung" zu nennen, welche sich in postiv und negativ spaltet. Würde „Lust" charakterisiert werden, würde es einen deutlich positiven Wert bekommen, jedoch keinen negativen. „Freude" hingegen hätte zwar auch keinen negativen Wert, allerdings einen geringeren positiven Wert. Ebenso verhält es sich mit Wut und Angst, nur auf der „nagativ"-Skala.

| **Bezug** | **Gegensätze, Extreme** | |
|---|---|---|
| **Objektbezug** | gerichtet (Moral,Ästhetik) | neutral (Stimmung) |
| **Richtung** | positiv (Lust,Freude) | negativ (Wut,Angst) |
| **Intensität** | stark | schwach |
| **Dauer** | anhaltend (Leidenschaft) | vergänglich (Affekt) |
| **Realität** | adäquat,angemessen | inadäquat |
| **Werte** | höhere,geistige,kulturelle (Kopf) | niedere,Leibliche (Bauch,Unterleib) |
| **Handeln** | fördernd | hemmend |
| **Teilnahme** | aktiv handelnd | passiv erlebend |

**Abbildung 56:** Charakteristika von Emotionen [Voe02]

Nachdem die grundlegenden Begriffe erläutert wurden, soll zuerst einmal ein Bogen von Emotion zu Verhalten gespannt werden, um den Zusammenhang zu beleuchten. Nachdem die Verknüpfung zwischen Emotion und Verhalten bekannt ist, stellt sich die Frage, wie Emotionen modelliert werden können, da Verhalten entscheidend davon abhängt, daher soll ein Ansatz zur Beschreibung von Emotionen erarbeitet werden, der Prototypenansatz.

Sind nun die Modellierungsgrundlagen gelegt, kann die Modellierung beginnen, jedoch existieren eine weite Menge von Emotionen und eine Einschränkung der emotionalen Vielfalt kann erfoderlich sein. Zum Ende des Dokuments soll noch Verhalten in der Gruppe, genauer soziale Interpendenz, in den Mittelpunkt rücken, bevor ein Fazit gezogen wird.

## I.2 Ein Modell für Emotion und Verhalten

Um die Wechselwirkung von Emotionen mit Verhalten verstehen zu können, soll hier ein funktionales Modell, aus dem Dokument [Voe02], vorgestellt werden, das beschreibt, wie die Emotionen zum Handeln bzw Verhalten beitragen und wie sich aus der Interaktion mit der Umwelt Emotionen ergeben. Siehe „Abbildung I.2“

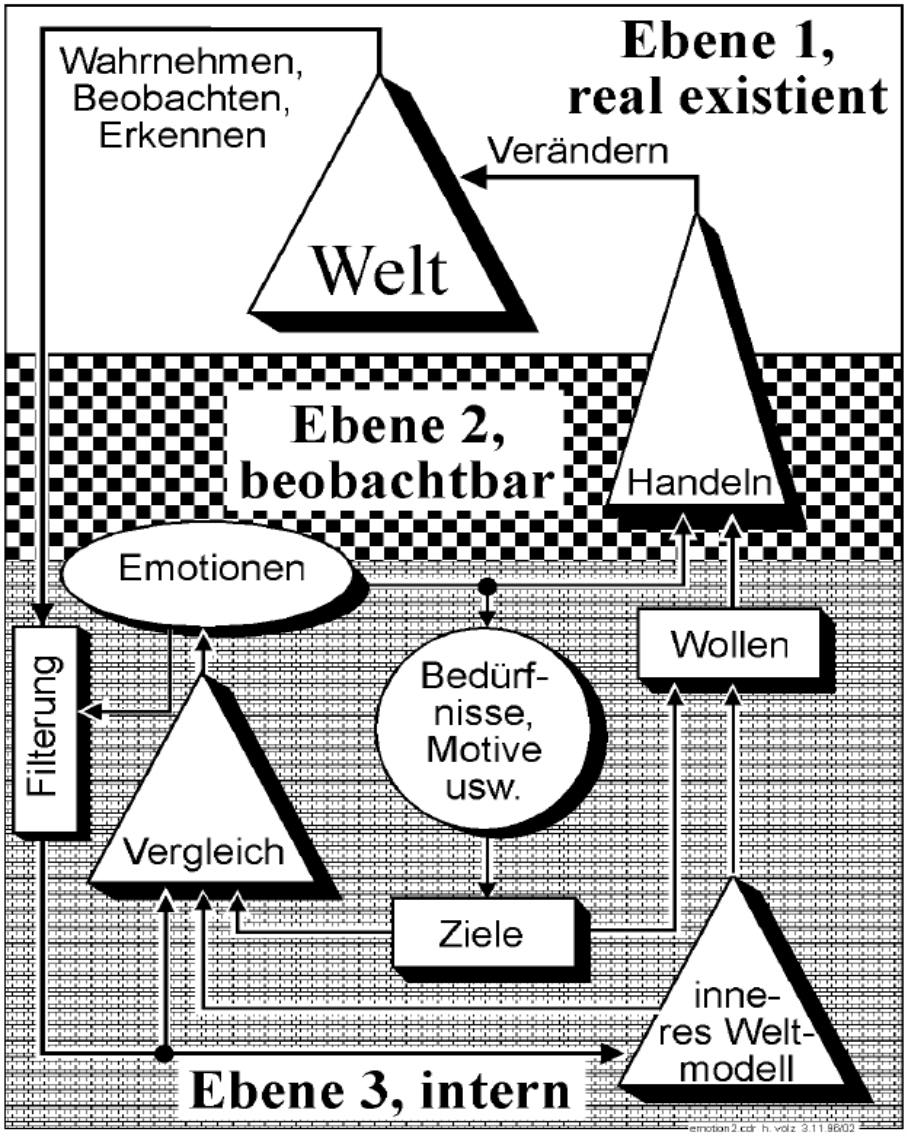



**Abbildung 57:** Modell für Emotionen und Verhalten [Voe02]

Grob besteht das Modell aus 3 Ebenen

1) der objektiven Umwelt
2) der sichtbaren Zustände, Aktionen des Subjekts
3) der internen Zustände des Subjekts

welche nun weitergehend erläutert werden.

Ebene 1
*Ebene 1 stellt die zugängliche objektive Umwelt, hier die Realität, dar, die vom Subjekt wahrgenommen und durch aktives Handeln verändert wird.*

Ebene 2
*Ebene 2 stellt das von außen beobachtbare Verhalten der Person dar, wie Handlung, oder Begleiterscheinungen von Emotionen (z.B. Tränen, Erröten).*

Ebene 3
*Ebene 3 sind die internen Zustände des Subjekts.*

Sie beschreiben ein spezielles inneres Modell, welches unter anderem Entstehung und Wirkung von Emotionen, Bedürfnisse und ein subjektives Weltmodell beschreibt. Das Subjekt nimmt die Umwelt, gefiltert durch Aufmerksamkeit und Interesse war, was in das innere Weltmodell einfließt. Beeinflusst durch den emotionalen Gemütszustand ergeben sich verschiedene Bedürfnisse, die in Zielen resultieren. Mittels des inneren Weltbilds, um die eigenen Ziele durchzusetzen, müssen aktive Handlungen gewollt und durchgeführt werden. Dieser Auswahlprozess geschieht durch Vergleiche bezüglich der Möglichkeiten und der unterschiedlichen Wichtigkeit von Zielen. Aus den Vergleichen entwickeln sich Emotionen, die bei gering erwartetem Aufwand positiv und bei schwierigen Aufgaben negativ ausfallen.

Aus dem obigen Modell ist klar zu erkennen, dass die Emotionen einen großen Einfluss auf das Verhalten und Handeln der Person haben. Somit wäre es essentiell zu wissen, wie eine Person auf Emotionen reagiert und wie man derartiges modellieren kann.
Eine gute Möglichkeit dazu bildet der Prototypenansatz.

## I.3 Der Prototypenansatz

Der Prototypenansatz besagt, dass Emotionen durch sogenannte emotionale Skripte ablaufen.
„Dabei handelt es sich um assoziative Wissensstrukturen, die verallgemeinerte Erfahrungen über den Ablauf emotionaler Erlebnisse beinhalten.“[BH04]

Dies bedeutet also, dass wir Wissen darüber haben, wie Emotionen ablaufen, aus der Kenntnis heraus, wie wir selbst Emotionen erleben, oder wie wir Emotionen von anderen Personen wahrnehmen. Dies ist in zweierlei Hinsicht nützlich, zum einen bilden sie eine Regieanweisung für eigenes Verhalten, zum anderen bilden sie ein Wahrnehmungsschema um Emotionen anderer zu erkennen. Durch die doppelte Funktionalität lässt sich die Emotion nicht von dem Wissen um die Emotion trennen.

Dadurch wird dieses Wissen in emotionalen Situationen automatisch genutzt. Eine solche Situation sieht nun folgendermaßen aus: „

1. emotionsauslösende Situationen (Ereignisse, kognitive Bewertungen und Befindlichkeiten, die die Entstehung von Emotionen begünstigen)
2. emotionale Reaktionen (Ausdrucksformen, Handlungsimpulse, physiologische Reaktionen, subjektive Empfindungen und typische Gedanken, die mit der Emotion

verbunden sind)

3. Selbstkontrolltechniken (Denk- und Verhaltensweisen, die dazu dienen, die emotionale Erregung zu regulieren bzw. Ausdrucks- und Handlungsimpulse zu unterdrücken)"[BH04]

Dies wird in Abbildung I.3 mit einer kleineren Verfeinerung, in der die emotionale Reaktion in zwei Aspekte unterteilt wird, grafisch dargestellt.

Diese Art Skripte sind natürlich den Eigenheiten der Kultur des Individuums, genauso wie dem Individuum selbst und weiteren Spezifika unterworfen. Trotzdem soll hier ein beispielhaftes Skript angegeben werden, dass wichtige Aspekte beinhaltet, es handelt sich hierbei um ein „Ärger"-Skript: „

- **Auslöser:** Ungerechtfertigte Unterbrechung, Regelverletzung oder Schädigung

  Etwas oder jemand widerspricht den Wünschen oder Erwartungen der Person, stört oder behindert die Bewegungsfreiheit oder die Erreichung von Zielen der Person, verletzt oder beleidigt die Person, ignoriert oder erniedrigt den Status der Person. Diese Störung oder Schädigung wird als ungerechtfertigt wahrgenommen, als etwas, das nicht passieren sollte und nicht zugelassen werden sollte.

- **Reaktionen:** Energischer Protest, Angriff, Rache

  Die Person wird energisch, ihre Gedanken und ihr Verhalten sind darauf ausgerichtet zu protestieren, zu kämpfen oder sich zu rächen, um auf diese Weise ihre Bewegungsfreiheit, die ihr zustehende Anerkennung usw. wiederherzustellen. Die Person sieht ärgerlich aus und klingt ärgerlich (z.B. rotes Gesicht, zusammengezogene Augenbrauen, laute Stimme) und bewegt sich nachdrücklich, einschüchternd oder übertrieben. Die Person ist gedanklich mit der Ärger auslösenden Situation beschäftigt und insistiert wiederholt, dass er oder sie im Recht ist, eine bessere Behandlung verdient usw.

- **Selbstkontrolltechniken:** Unterdrückung und Umdeutung

  Die Person kann versuchen, den Ärger zu unterdrücken oder zu verbergen oder die Situation so umzudeuten oder zu verändern, dass sie keinen Anlass zum Ärger mehr gibt."[BH04]

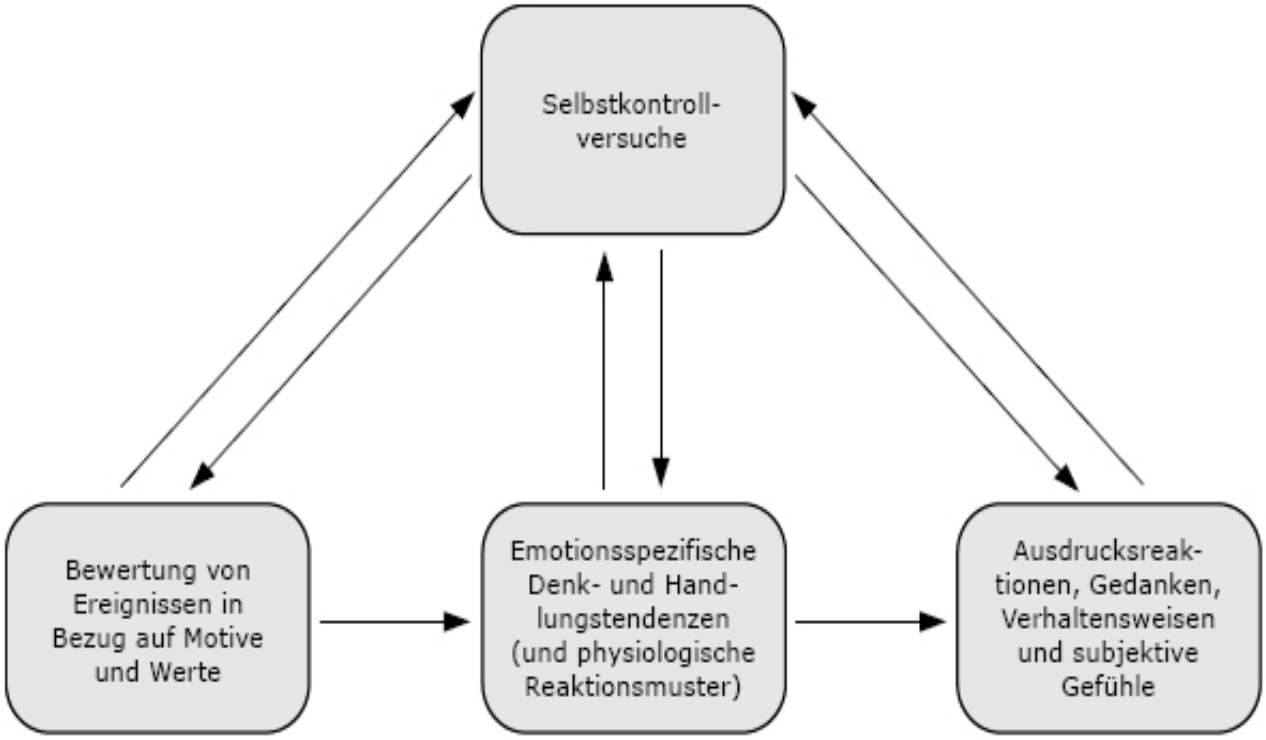



**Abbildung 58:** Aufbau einer emotionalen Situation [BH04]

## I.4 Einschränkung der emotionalen Vielfalt

Es besteht ein breit gefächertes Spektrum an Emotionen und eine Vielzahl an Emotionsbegriffen, die zu modellieren sind. Daraus könnte sich also die Notwendigkeit entwickeln, die Zahl an Emotionen zu verringern. Viele Psychologen teilen Emotionen in Primäremotionen und Sekundaremotionen auf. Sie greifen also einige Emotionen heraus und stellen diese als übergeordnet dar. Allerdings wird kritisiert, dass diese Aufteilungen nicht einheitlich (vgl [OT90]) sind und deshalb sollen hier Herangehensweisen vorgestellt werden, die zur Begründung einer Wahl auf einzelne Emotionen dienen könnten.

Ein Ansatz wäre es, die Emotionen in Dimensionen einzuteilen. Osgood, Suci und Tannenbaum (1957) [OS57] ordneten diese in drei Dimensionen

- Evaluation (angenehm - unangenehm)
- Erregung (beruhigend - erregend)
- Potenz (stark - schwach)

Hierzu könnte auch die schon vorgestellte Tabelle „Abbildung I.4“ als Ausgangspunkt dienen.

Nun können die einzelnen Emotionen in das Koordinatensystem eintragen werden, nachdem festgelegt wurde, wie die Emotion in den einzelnen Dimensionen zu bewerten sind, abhängig von der Platzierung anderer Emotionen. Beispielsweise ist Ärger in der Dimension Potenz recht „stark“, aber wertetechnisch nicht so hoch angelegt, wie die Wut die eine noch „stärkere“Emotion darstellt. Im Zuge einer Vergröberung könnten nun einige Emotionen zusammengefasst werden.

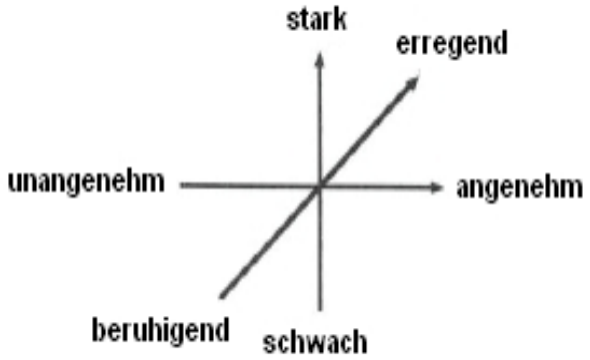



**Abbildung 59:** Dimensionen für Emotionen [Fun07]

Das Problem an dieser Vorgehensweise besteht darin, dass es wieder keine einheitliche Festlegung der Dimensionen gibt und somit viele verschiedene Modelle bestehen können.

Eine leicht andere herangehensweise ist es, eine Clusteranalyse über die Emotionen durchzuführen. Dies wurde anhand einer Untersuchung (Schmidt-Atzert 1980) durchgeführt, welche durch eine Auswahl von Emotionen aus einem Pool von Emotionsbegriffen, die nach Angabe der Probanden am ehesten Emotionen beschrieben, einer Klassifizierung und anschliessenden Erstellung einer Ähnlichkeitsmatrix hergeführt wurde. Das Ergebnis, nach einer clusteranalytischen Auswertung, sah dann folgendermaßen aus:

Je näher die Verknüpfung der Balken einzelner Emotionen, oder Emotionsgruppen, an ihrer Betitelung stattfindet (also je „weiter links“), umso ähnlicher wurden diese Emotionen empfunden. Diese Verknüpfungen fügen einzelne Emotionen zu Gruppen zusammen. Wenn nun ein senkrechter Schnitt im Diagramm durchgeführt wird, erhält man Emotionsgruppen, welche aus ähnlichen Emotionen bestehen. Wobei hier gilt, dass je weiter „gegen 0“ der Schnitt gezogen wird, umso mehr Gruppen resultieren, jedoch auch nur Gruppen, deren Emotionen sehr nah „verwandt“ sind.

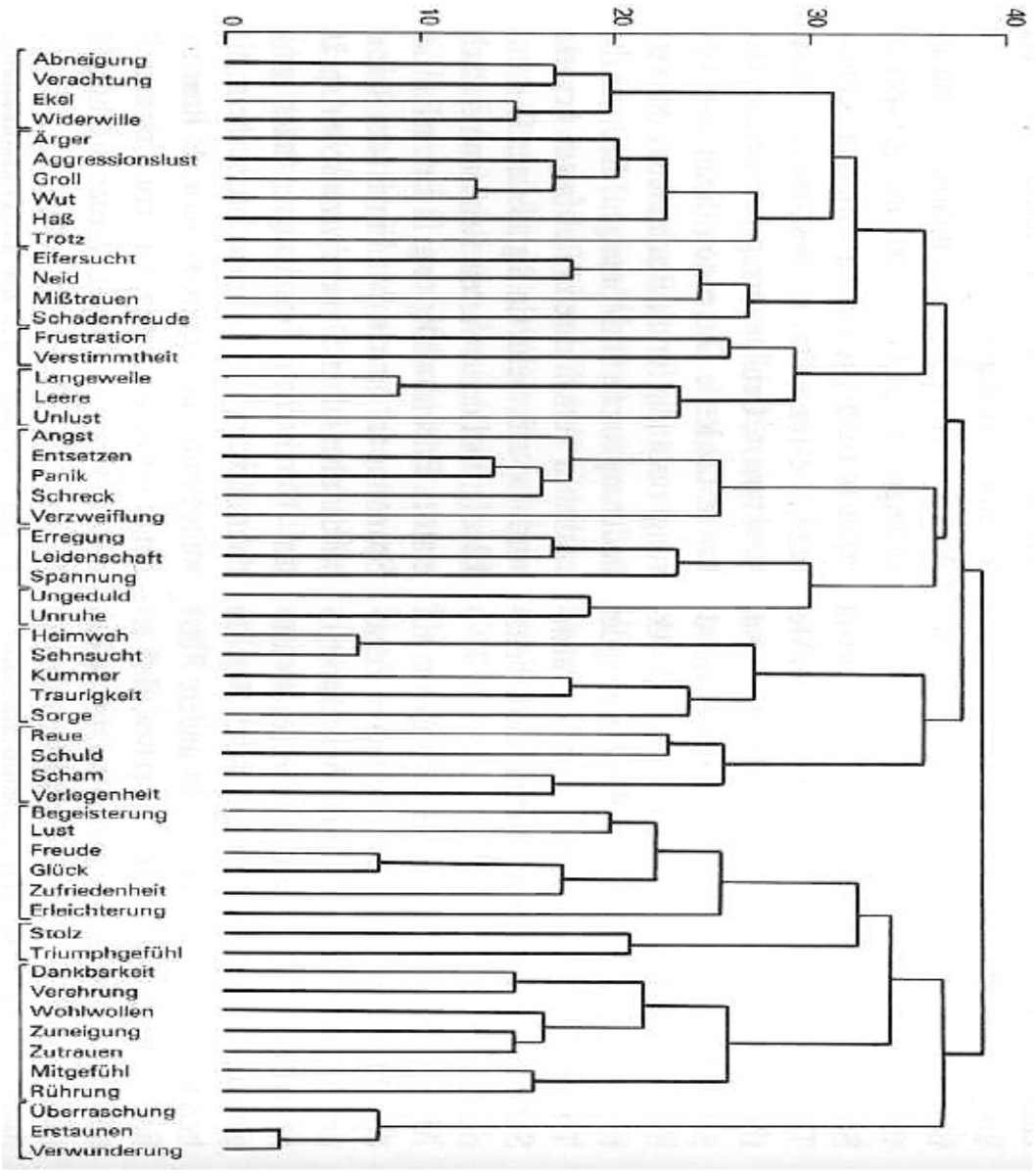



**Abbildung 60:** Clusteranalyse von Emotionen [Fun07]

## I.5 Soziale Interpendenztheorie

Betrachtet wurden bisher individuelle Emotionen und dazugehöriges Handeln. Nun soll beleuchtet werden, welche Eigenheiten in Gruppen bezüglich des Verhaltens gelten.

Zuerst sollte der Begriff Interdependenz erläutert werden.

### Definition Interpendenz

„Wechselseitige Abhängigkeit und Einflussnahme von Personen in Interaktionen und Beziehungen."[AFM02]

Soziale Interpendenz besteht also dann, wenn das Ergebnis des Einen, von den Aktionen des Anderen beeinflusst wird, und dies bidirektional. Somit existieren zwei Arten von sozialer Interpendenz: kooperativ und kompetitiv. Abwesenheit sozialer Interpendenz resultiert in individualistischem Bestreben.
Es bestehen also drei Möglichkeiten der Verbindung einer Aktion eines Individuums zu den Aktionen der Anderen: „

1. **kooperative Zusammenarbeit um gemeinsame Ziele zu erreichen**
   Bei einer kooperativ strukturierten Aufgabe sind individuelle Ziele positiv korreliert. Dies bedeutet, dass das individuelle Ziel nur erreicht werden kann, sofern die anderen Gruppenmitglieder auch ihr Ziel erreichen. Also suchen die Individuen Ergebnisse, die Vorteilhaft für alle kooperativen Partner sind.

2. **kompetitives Verhalten bei Zielen, dass nur wenige erreichen können**
   In einer kompetitiven Situation arbeiten Individuen gegeneinander, um ein Ziel zu erreichen, dass nur einer, oder wenige erreichen können. Die individuellen Ziele sind somit negativ korreliert. Jedes Individuum erkennt, dass wenn eine Person ihr Ziel erreicht, alle Anderen, mit denen diese Person kompetitiv verbunden ist, ihr Ziel verfehlen. Also versuchen die Individuen Ergebnisse zu produzieren die ihnen persönlich vorteilhaft sind, aber situationsbedingt abträglich den Anderen.

3. **individuelles Handeln bei unverknüpften Zielen bezüglich anderen Individuen**
   Wenn eine Situation individualistisch ausgerichtet ist, gibt es keine zieltechnische Korrelation zu anderen Teilnehmern. Jeder verdient das, was er erreichen kann unbeachtet davon, ob andere Individuen ihr Ziel erreichen, oder nicht. Daraus folgt, dass sich das Verhalten am persönlichen Ziel ausrichtet, ohne auf andere Ergebnisse einzugehen.“[JJ98]

Positive Interpendenz führt also zu begünstigendem Verhalten, genauso wie negative zu gegensätzlichem Verhalten. Bei negativer Interpendenz versucht also jeder seine eigenen Ziele zu erreichen und schadet somit der Zielerreichung seiner Konkurrenten, während bei unverknüpften Zielen letztere Komponente wegfällt. Nun sollte ein Blick auf das begünstigende Verhalten gelegt und dies näher beschrieben werden.

Begünstigendes Verhalten zeichnet sich hierbei durch folgende Taten aus: „

1. Hilfe leisten und bekommen
2. Austausch von Ressourcen und Informationen
3. Feedback zur Arbeitsweise, um Performanz zu verbessern
4. kritisches Prüfen der Begründung von Anderen (z.B. bei kooperativem Lernen)
5. suche nach neuen (im Sinne der Aufgabenstellung guten) Zielen
6. beeinflussen und beeinflussen lassen von anderen Kollegen (das bessere Lösungsverfahren setzt sich durch)
7. nutzen von sozialem Geschick, für effektives Teamwork
8. Prüfung der Effektivität einzelner Gruppenmitglieder und Verbesserung der Gruppenperfomanz“ (übersetzt)[JJ98]

Hierbei ist soziales Geschick (Punkt7) von besonderer Bedeutung, da soziales Geschick die Kooperation zwischen den einzelnen Mitgliedern fördert. Wenn es also erforderlich ist zusammenzuarbeiten, müssen die Personen „

(1) sich kennenlernen und vertrauen

(2) eindeutig und akurat miteinander kommunizieren

(3) Akzeptanz und Unterstützung untereinander haben

(4) Konflikte konstruktiv lösen“ (übersetzt)[JJ98]

## I.6 Fazit

In der Psychologie und Soziologie gibt es einige interessante Theorien über Emotionen und Verhalten. Es wurden sowohl ein Modell für Emotionen, als auch die Verbindung mit Verhalten erläutert. Weiterhin wurde aufgezeigt, wie die Anzahl der zu modellierenden Emotionen verringert werden kann und das Verhalten in Gruppen näher erläutert. Hier ist jedoch im Hinblick auf die Aufgabe der PG, menschliche wirkende computergesteuerte Spielercharaktere zu entwerfen, festzustellen, dass diese verständlicherweise nicht dazu ausgelegt wurden, um auf einem Computer zum Einsatz zu kommen, jedoch ist dies nicht grundsätzlich ausgeschlossen. Auch ist anzumerken, dass dies nur ein kleiner Ausschnitt, aus der Welt der Psychologie und Soziologie, war.

Ich denke, daher, dass die psychologische und soziologische Herangehensweise eine gute Basis darstellt, im Hinblick auf die Zielsetzung, der Modellierung eines menschlich wirkenden Computerspielers. In den meisten Spielen werden psychologische Effekte meist nur angewendet, um den Spieler zu animieren weiterzuspielen, aber selten enthalten künstliche Intelligenzen ein menschliches Verhalten. Dies ließe sich mit den hier vorgestellten Methoden durchaus bewerkstelligen, um Menschen ein noch größeres Interesse an künstlichen Intelligenzen zu vermitteln und den Spielspaß zu erhöhen.

# J PG-Ordnung

## J.1 PG-Ordnung

### J.1.1 In der PG-Ordnung sind verbindliche Pflichten für alle PG-Teilnehmer enthalten.

### J.1.2 Abstimmungsmodus

- Satzungsänderungen:
  Können nur beschlossen werden wenn mindestens zwei Drittel der Teilnehmer für die Änderung stimmen (qualifizierte Mehrheit).
- Alle anderen Änderungen:
  Änderungen werden per Mehrheitsentscheid (einfache Mehrheit) getroffen.
- Eine Abstimmung findet nur dann statt, wenn bezüglich eines Themas Kontroversen bestehen.
- Alle durchgeführten Abstimmungen sind offen.

### J.1.3 Organisatorisches

- Anwesenheit:
  Die Teilnahme an den PG-Sitzungen ist Pflicht. Im Protokoll wird vermerkt, wenn ein PG-Teilnehmer oder ein Betreuer verspätet zur PG-Sitzung eintrifft.
- Beginn der PG-Sitzungen:
  PG-Stizungen starten grundsätzlich pünktlich, auch wenn PG-Teilnehmer oder PG-Betreuer fehlen.

## J.2 Etikette

### J.2.1 In der Etikette sind alle wünschenswerten Aufgaben enthalten.

- Sowohl E-Mails als auch das Wiki müssen einmal am Werktag überprüft werden.
- Der Protokollant soll das Protokoll einen Tag nach der Sitzung bereitstellen.
- Der Moderator soll die Tagesordnung einen Tag vor der Sitzung bereitstellen.
- Ist ein PG-Teilnehmer verhindert, so muss dies den Betreuern und den restlichen PG-Teilnehmern per E-Mail und mit Angabe eines triftigen Grunds mitgeteilt werden.
- E-Mails sollen mit dem Ausdruck „[PG529]“ beginnen.
- Eine förmliche Anrede ist nicht nötig.
- Eine Änderung der Etikette bedarf einer einfachen Mehrheit.

# K Fragen der Umfrage inkl. Antwortmöglichkeiten

## K.1 Fragen und Antwortmöglichkeiten der Umfrage

1. Wie alt sind Sie?
   - keine Angabe
   - $\leq 20$
   - 21-25
   - 26-30
   - 31-35
   - 36-40
   - 41-50
   - 51-60
   - >60
2. Bitte geben Sie Ihr Geschlecht an!
   - keine Angabe
   - Frau
   - Mann
3. Bitte geben Sie Ihren höchsten erreichten Schulabschluss an!
   - keine Angabe
   - Hauptschulabschluss
   - mittlere Reife
   - Abitur
   - Hochschulabschluss
4. Haben Sie schon einmal gespielt?
   - keine Angabe
   - Ja
   - Nein
5. Wie lange spielen sie schon Poker?
   - keine Angabe
   - über 3 Jahre
   - seit 2-3 Jahren
   - seit 1 Jahr
   - 6-12 Monate
   - kürzer
6. Wie oft spielen Sie Poker am Computer?

- keine Angabe
- mehrmals pro Woche
- wöchentlich
- 1-2 mal pro Monat
- 1-2 mal im Jahr
- noch seltener

7. Wie häufig spielen Sie direkt gegen Menschen?
   - keine Angabe
   - mehrmals pro Woche
   - wöchentlich
   - 1-2 mal pro Monat
   - 1-2 mal im Jahr
   - noch seltener
8. Wie gut würden Sie sich als Pokerspieler einschätzen?
   - keine Angabe
   - Profi
   - Fortgeschrittener
   - Anfänger
9. Welcher Gegenspieler ist Ihnen am liebsten?
   - keine Angabe
   - mit Gefühlen und Schwächen
   - mit Gefühlen und Spielstärke
   - hauptsache spielstark
10. Spielen Sie lieber Poker um des Spiels wegen oder um Geld zu gewinnen?
    - keine Angabe
    - vorderrangig des Spiels wegen
    - Beides ist mir wichtig
    - vorderrangig um Geld zu gewinnen
11. Spielen Sie lieber gegen passive oder aggressive Gegner?
    - keine Angabe
    - gegen passive
    - teils teils
    - gegen aggressive
12. Hat Ihnen das Spielen Spass gemacht?
    - keine Angabe
    - ja

- ein wenig
- neutral
- eher nicht
- nein

13. Haben Sie Nachrichten von Ihren Gegenspielern wahrgenommen?
    - keine Angabe
    - Ja
    - Nein
14. Fanden Sie die Kommunikation mit Ihren Mitspielern sinnvoll?
    - keine Angabe
    - ja
    - ein wenig
    - ich bin mir nicht sicher
    - eher nicht
    - nein
15. Haben Sie Nachrichten verschickt?
    - keine Angabe
    - ja regelmäßig
    - öfters
    - ab und zu
    - eher weniger
    - gar nicht
16. Wie gut fanden Sie den Einsatz der Emoticons?
    - keine Angabe
    - sehr gut
    - gut
    - neutral
    - weniger gut
    - schlecht
17. Welcher Gegner war am menschenähnlichsten?
    - keine Angabe
    - Bernd
    - Gerd
    - Markus
    - Karsten
18. Welchen Gegner empfanden Sie am spielstärksten?

- keine Angabe
- Bernd
- Gerd
- Markus
- Karsten

19. Wie anspruchsvoll haben die Gegner gespielt?
    - keine Angabe
    - zu schwer
    - eher zu schwer
    - gerade richtig
    - eher zu leicht
    - zu leicht
20. Konnten Sie unter den Spielern Unterschiede in der Spielweise feststellen?
    - kein Angabe
    - ja
    - ein wenig
    - ich bin mir nicht sicher
    - eher nicht
    - nein
21. Stellten die Spieler einen - über den Spielverlauf hinweg - kontinuierlichen Charakter dar?
    - keine Angabe
    - ja
    - ein wenig
    - ich bin mir nicht sicher
    - eher nicht
    - nein
22. Hatten Sie das Gefühl, dass einer der Gegenspieler geblufft hat?
    - keine Angabe
    - ja
    - ein wenig
    - ich bin mir nicht sicher
    - eher nicht
    - nein

# L Spielertypen des Opponent Modelings

Tabelle der Spielertypen:

| | player type | VPIP | PFR | Post-flop aggression |
|---|---|---|---|---|
| 1 | Loose Passive/Passive | 30+ | <5 | <1.5 |
| 2 | Loose Passive/Aggressive | 30+ | <5 | >2 |
| 3 | Semi-Loose Passive/Passive | 20-30 | <5 | <1.5 |
| 4 | Semi-Loose Passive/Aggressive | 20-30 | <5 | >2 |
| 5 | Loose Aggressive/Passive | 30+ | 10+ | <2 |
| 6 | Loose Aggressive/Aggressive | 30+ | 10+ | >2.5 |
| 7 | Semi-Loose Aggressive/Passive | 20-30 | 10+ | <2 |
| 8 | Semi-Loose Aggressive/Aggressive | 20-30 | 10+ | >2 |
| 9 | Tight Passive/Passive | <20 | <5 | <1.5 |
| 10 | Tight Passive/Aggressive | <20 | <5 | >2 |
| 11 | Tight Aggressive/Aggressive | <20 | >10 | >2.5 |
| 12 | Tight Aggressive/Passive | <20 | >10 | <2 |

# M Verknüpfung der Charakterwerte mit den Parametern (Opponent Modeling)

| | Position | Wartezeit | Raise-Häufigkeit | Raise-Beträge |
|---|---|---|---|---|
| Ehrlichkeit | | -0.2 | | |
| Aggressivität | 1.0 | -0.2 | 2.5 | 3.0 |
| Risikobereitschaft | -1.0 | | | |
| Zögerlichkeit | 0.8 | 4.0 | | |
| Blattbewertung | | -1.0 | | |
| Kartenbewertung | | | | |

| | Ausstiegsphase | Cash |
|---|---|---|
| Ehrlichkeit | -1.0 | 1.2 |
| Aggressivität | | -0.5 |
| Risikobereitschaft | 3.0 | -2.0 |
| Zögerlichkeit | | 1.0 |
| Blattbewertung | | cash |
| Kartenbewertung | | 1.0 |

| | Pot | Phase | Gemeinschaftskarten |
|---|---|---|---|
| Ehrlichkeit | | | |
| Aggressivität | | | 1.0 |
| Risikobereitschaft | 1.0 | 0.5 | -0.5 |
| Zögerlichkeit | 1.0 | | |
| Blattbewertung | | | |
| Kartenbewertung | | | -0.2 |

# N Quellcode

## N.1 C++ Code zur Veränderung der Emotionen

```
/**
 * Veraendert die Emotionen des Spielers m_player, wenn er eine Nachricht erhaelt.
 * Alle Informationen, die benoetigt werden, werden vom EmoPlayer geholt.
 * Ueberschreibt die Funktion aus EmoStrategy.
 * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von der Nachricht und den Charakterwerten des Spieler abhaengen.
 * \param gegner          Von welchem Spieler wurde die Nachricht gesendet
 * \param gerichtet       true gdw. die Nachricht an den Spieler gerichtet ist (nicht an den Tisch)
 * \param nachricht       Nummer der Nachricht
 * \return                Nothing
 */
void DeterministischeEmoAnpassung::recieveMessage(int gegner, bool gerichtet, int nachricht){
        EmoPlayer* player = getPlayer ( );
        //int action = player -> getMyAction(); // sollte 1 sein, wenn ich gefoldet habe.
        bool fold = player -> getGefoldet(); //true, wenn gefoldet

        player->calcMyOdds();

        //Anzahl Spieler
        int players = player -> getNumberOpponents();
        if(players > 5) players = 5;
        if(players < 2) players = 2;
        players = players -2;

        //Fehlerrichtung
        srand( (unsigned) time(NULL));
        int fehlerrichtung = rand() % 2;
        if (fehlerrichtung == 0) fehlerrichtung = -1;

        //subjektive Schaetzung der Hand
        double schaetzung = (100 - player->getCharacter_kartenBewerten())
            / 8 * fehlerrichtung + player->getMyOdds();

        int zufriedenheit = 1;  //zufriedenheit = 2 : Karten sind super
                                //zufriedenheit = 0 : Karten sind scheisse
                                //zufriedenheit = 1 : sonst

        switch(player->getCurrentRound()){
                case 0: //preflop
                        switch(players){
                                case 0: //85-35
                                        { if (schaetzung > 73)
                                            zufriedenheit = 2;
                                          else if (schaetzung < 47)
                                            zufriedenheit = 0;
                                        }break;
                                case 1: //73-22
                                        { if (schaetzung > 60)
                                            zufriedenheit = 2;
                                          else if (schaetzung < 35)
                                            zufriedenheit = 0;
                                        }break;
                                case 2: //64-15
                                        { if (schaetzung > 52)
                                            zufriedenheit = 2;
                                          else if (schaetzung < 27)
                                            zufriedenheit = 0;
```

```
50                                      }break;
51                              case 3: //56-12
52                                      { if (schaetzung > 45)
                                            zufriedenheit = 2;
53                                        else if (schaetzung < 23)
                                              zufriedenheit = 0;
54                                      }break;
55                      }
56                      break;
57              case 1: //flop
58                      switch(players){
59                              case 0: //93-16 = 77
60                                      {  if (schaetzung > 74)
                                            zufriedenheit = 2;
61                                         else if (schaetzung < 35)
                                               zufriedenheit = 0;
62                                      }break;
63                              case 1: //87-8 = 70
64                                      {  if (schaetzung > 70)
                                            zufriedenheit = 2;
65                                         else if (schaetzung < 25)
                                               zufriedenheit = 0;
66                                      }break;
67                              case 2: //82-4 = 78
68                                      {  if (schaetzung > 63)
                                            zufriedenheit = 2;
69                                         else if (schaetzung < 23)
                                               zufriedenheit = 0;
70                                      }break;
71                              case 3: //78-3 = 75
72                                      {  if (schaetzung > 59)
                                            zufriedenheit = 2;
73                                         else if (schaetzung < 22)
                                               zufriedenheit = 0;
74                                      }break;
75                      }
76
77                      break;
78              default: //sonst
79                      if (schaetzung > 75) zufriedenheit = 2;
80                      else if (schaetzung < 25) zufriedenheit = 0;
81                      break;
82      }
83
84      //Emotionen vom Spieler holen
85      vector<int> wut = player -> getEmotion_hass();
86      int freude = player -> getEmotion_trauerEuphorie();
87      int nervositaet = player -> getEmotion_selbstsicherheitPanik();
88
89      //Anpassen der Emotionen nur, wenn gegner ein gueltiger wert ist
90      if(gegner > 0 && gegner < wut.size()){
91              //Nachrichten auswerten
92              if(nachricht >= 0 && nachricht <= 4){           //freude
                    gerichtet
93                      if(!fold){
94                              nervositaet = nervositaet + (3 -
                                    zufriedenheit);
95                      }
96                      if(nachricht == 3){     //Du spielst wie ein
                            Anfaenger.
97                              wut[gegner] = wut[gegner] + 2;
98                              if (zufriedenheit == 2){
99                                      freude = freude + 1;
100                             }
101                     }
102             }else if(nachricht >= 5 && nachricht <= 18){    //freude
                    ungerichtet
103                     if(!fold){
104                             if (nachricht <= 8){
```

```
105                                         nervositaet = nervositaet + (3 -
                                                zufriedenheit);
106                                 }else{
107                                         nervositaet = nervositaet + (3 -
                                                zufriedenheit) / 2;     //bei
                                                super Karten keine erhoehung
108                                 }
109                         }
110                 }else if(nachricht >= 19 && nachricht <= 24){   //
                        schadenfreude gerichtet
111                         freude = freude - 1;
112                         wut[gegner] = wut[gegner] + 1;
113                 }else if(nachricht >= 25 && nachricht <= 33){   //wut
                        gerichtet
114                         wut[gegner] = wut[gegner] + 5;
115                         if (zufriedenheit != 0 & !fold){
116                                 nervositaet = nervositaet - 1;
117                         }
118                 }else if(nachricht >= 34 && nachricht <= 49){   //wut
                        ungerichtet
119                         if (zufriedenheit != 0 & !fold){
120                                 nervositaet = nervositaet - 2;
121                         }
122                 }else if(nachricht >= 50 && nachricht <= 50){   //
                        nervositaet gerichtet
123                         wut[gegner] = wut[gegner] - 1;
124                         if(!fold){
125                                 nervositaet = nervositaet - 2 -
                                        zufriedenheit;
126                         }else if (zufriedenheit == 2){
127                                 freude = freude +1;
128                         }
129                 }else if(nachricht >= 51 && nachricht <= 59){   //
                        nervositaet ungerichtet
130                         if(!fold){
131                                 nervositaet = nervositaet - 1 -
                                        zufriedenheit;
132                         }
133                         if (nachricht >= 52){
134                                 wut[gegner] = wut[gegner] - 1;
135                                 if (zufriedenheit == 2 && !fold){
136                                         freude = freude +1 ;
137                                 }
138                         }
139                 }
140
141                 //Emotionen neu setzen
142                 player -> setNeutralHassPlayer(gegner, wut[gegner]);
143                 player -> setTrauerEuphorie ( freude );
144                 player -> setSelbstsicherheitNervositaet ( nervositaet );
145         } else {
146                 cout << "Parameter gegner in der Funktion recieveMessage
                        in DeterministischeEmoAnpassung war ungueltig" << endl
                        ;
147                 exit(1);
148         }
149 }
150
151 /**
152  * Veraendert die Emotionen des EmoPlayers m_player, wenn dieser seine
        Holecards erhaelt.
153  * Alle Informationen, die benoetigt werden, werden vom EmoPlayer geholt.
154  * Ueberschreibt die Funktion aus EmoStrategy.
155  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
156  * \return Nothing
157  */
158 void DeterministischeEmoAnpassung::recieveHoleCards(){
159         EmoPlayer* player = getPlayer ( );
```

```
160
161         player->calcMyOdds();
162
163         //Anzahl Spieler
164         int players = player -> getNumberOpponents();
165         if(players > 5) players = 5;
166         if(players < 2) players = 2;
167         players = players -2;
168
169         //Fehlerrichtung
170         srand( (unsigned) time(NULL));
171         int fehlerrichtung = rand() % 2;
172         if (fehlerrichtung == 0) fehlerrichtung = -1;
173
174
175         //Chancen players = 0: 85-35=50
176         //Chancen players = 1: 73-22=51
177         //Chancen players = 2: 64-15=49
178         //Chancen players = 3: 56-12=44
179
180         //subjektive Schaetzung der Hand
181         double schaetzung = (100 - player->getCharacter_kartenBewerten())
                / 8 * fehlerrichtung + player->getMyOdds();
182
183         // Berechne das Potenzial, wie sehr er sich freuen kann (oder eben
                nicht)
184         // der durchschnittliche Spieler freut sich, wenn seine Hand im
                oberen Bereich der W'keiten liegt
185         // hier das setzverhalten Haende wichtig
186
187         //je hoeher sertzverhaltenHaende, desto schlechtere Blaetter
                spielt er
188         double neutralesPotential = schaetzung + (player->
                getCharacter_setzverhaltenHand() - 50) / 12;
189         if (neutralesPotential <0) neutralesPotential = 0;
190         if (neutralesPotential >100) neutralesPotential = 100;
191
192         double freude;
193         switch(players){
194                 case 0: { freude = neutralesPotential - 60;
195                         }break;
196                 case 1: { freude = neutralesPotential - 48;
197                         }break;
198                 case 2: { freude = neutralesPotential - 39;
199                         }break;
200                 case 3: { freude = neutralesPotential - 31;
201                         }break;
202                 default: freude = 0;
203         }
204
205         //freude ist im Bereich von -41,666666 bis 41,666666
206
207         double normierteFreude = freude / 41;    //kann auch negativ sein
                !!!!
208
209         int emotionFreude = player->getEmotion_trauerEuphorie() +
                normierteFreude * 4;
210
211         //historyfunktion
212         holecardFreude[holecardFreudeZeiger] = normierteFreude;
213
214         int i; int durchschnitt = 0; int nenner = 0;
215
216         if(holecardFreudeVoll){
217                 holecardFreudeZeiger = (holecardFreudeZeiger + 1) %
                        holecardFreudeGroesse;
218                 for(i =0 ; i < holecardFreudeGroesse; i++){
219                         durchschnitt = durchschnitt + holecardFreude[ (
                                holecardFreudeZeiger + i) %
```

```
                                holecardFreudeGroesse ]*(i+1);
220                             nenner = nenner + i + 1;
221                     }
222             } else {
223                     if(++holecardFreudeZeiger == holecardFreudeGroesse){
224                             holecardFreudeVoll = true;
225                             //holecardFreudeZeiger = 0;
226                     }
227                     for(i =0; i < holecardFreudeZeiger; i++){
228                             durchschnitt = durchschnitt + holecardFreude[i]*(i
                                    +1);
229                             nenner = nenner + i + 1;
230                     }
231                     if(holecardFreudeVoll)
232                             holecardFreudeZeiger = 0;
233             }
234             durchschnitt = durchschnitt / nenner;
235
236             emotionFreude = emotionFreude + durchschnitt * 4;
237
238             player->setTrauerEuphorie(emotionFreude);
239     }
240
241     /**
242      * Veraendert die Emotionen des Spielers m_player, wenn neue
              Communitycards auf den Tisch kommen.
243      * Alle Informationen, die benoetigt werden, werden vom EmoPlayer geholt.
244      * Ueberschreibt die Funktion aus EmoStrategy.
245      * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
              den Charakterwerten des Spieler und den Karten abhaengen.
246      * \return Nothing
247      */
248     void DeterministischeEmoAnpassung::recieveCommunityCards(){
249             EmoPlayer* player = getPlayer ( );
250             //int action = player -> getMyAction(); // sollte 1 sein, wenn ich
                      gefoldet habe.
251             bool fold = player -> getGefoldet(); //true, wenn gefoldet
252
253             double myOldOdds = player->getMyOdds();
254             //Fehlerrichtung
255             srand( (unsigned) time(NULL));
256             int fehlerrichtung = rand() % 2;
257             if (fehlerrichtung == 0) fehlerrichtung = -1;
258             //subjektive Schaetzung der Hand
259             double schaetzungAlt = (100 - player->getCharacter_kartenBewerten
                    ()) / 8 * fehlerrichtung + myOldOdds;
260
261             player->calcMyOdds();
262             double myOdds = player->getMyOdds();
263             //Fehlerrichtung
264             fehlerrichtung = rand() % 2;
265             if (fehlerrichtung == 0) fehlerrichtung = -1;
266             //subjektive Schaetzung der Hand
267             double schaetzung = (100 - player->getCharacter_kartenBewerten())
                    / 8 * fehlerrichtung + myOdds;
268
269             //Anzahl Spieler
270             int players = player -> getNumberOpponents();
271             if(players > 5) players = 5;
272             if(players < 2) players = 2;
273             players = players -2;
274
275             double oddsDiff = schaetzung - schaetzungAlt;
276             oddsDiff = oddsDiff / 5;
277
278             int oddsMitte = 50;
279             // Ab wann ist Spieler mit Karten zufrieden
280             //je hoeher sertzverhaltenHaende, desto schlechtere Blaetter
                    spielt er
```

```
281         switch(player->getCurrentRound()){
282                 case 0: //preflop
283                         switch(players){
284                                 case 0: { oddsMitte = 60+(50-player->
                                        getCharacter_setzverhaltenHand())/12 ;
285                                         }break;
286                                 case 1: { oddsMitte = 48+(50-player->
                                        getCharacter_setzverhaltenHand())/12;
287                                         }break;
288                                 case 2: { oddsMitte = 39+(50-player->
                                        getCharacter_setzverhaltenHand())/12;
289                                         }break;
290                                 case 3: { oddsMitte = 31+(50-player->
                                        getCharacter_setzverhaltenHand())/12;
291                                         }break;
292                         }
293                         break;
294                 case 1: //flop
295                         switch(players){
296                                 case 0: //93-16 = 77 --> 54,5
297                                         { oddsMitte = 55+(50-player->
                                                getCharacter_setzverhaltenHand
                                                ())/12;
298                                         }break;
299                                 case 1: //87-8 = 70 --> 52
300                                         { oddsMitte = 52+(50-player->
                                                getCharacter_setzverhaltenHand
                                                ())/12;
301                                         }break;
302                                 case 2: //82-4 = 78 --> 43
303                                         { oddsMitte = 43+(50-player->
                                                getCharacter_setzverhaltenHand
                                                ())/12;
304                                         }break;
305                                 case 3: //78-3 = 75 --> 40
306                                         { oddsMitte = 40+(50-player->
                                                getCharacter_setzverhaltenHand
                                                ())/12;
307                                         }break;
308                         }
309
310                         break;
311                 default: //sonst
312                         oddsMitte = 50+(50-player->
                                getCharacter_setzverhaltenHand())/12 ;
313                         break;
314         }
315
316         //Emotionen vom Spieler holen
317         vector<int> wut = player -> getEmotion_hass();
318         int freude = player -> getEmotion_trauerEuphorie();
319         int nervositaet = player -> getEmotion_selbstsicherheitPanik();
320         int erhoehungGlobaleWut=0;
321
322         //Emotionen veraendern
323         // gewinnw'keit hat sich verbessert
324         if( oddsDiff > 0){
325                 if( fold ){ //ichHabeGefoldet
326                         if (myOdds > oddsMitte)
327                                 wut[0] = wut[0] + oddsDiff;
328                                 erhoehungGlobaleWut = oddsDiff;
329                 } else{
330                         freude = freude + oddsDiff;
331
332                         if (myOdds < oddsMitte)
333                                 nervositaet = nervositaet + (oddsMitte -
                                        myOdds) /8;
334                 }
335
```

```
336         // gewinnw'keit hat sich verschlechtert
337         }else    if ( fold ){ //ichHabeGefoldet
338                         freude = freude - oddsDiff;             //erhoehen
                                , weil oddsDiff neg.
339                         nervositaet = nervositaet + oddsDiff;   //
                                nervositaet veringern
340                 }else{
341                         freude = freude + oddsDiff;             //
                                veringern, weil oddsDiff neg.
342                         nervositaet = nervositaet - oddsDiff/2;//
                                nvervositaet steigt
343
344                         if (myOdds < oddsMitte)
345                                 nervositaet = nervositaet + (oddsMitte -
                                        myOdds) /8;
346                 }
347
348
349         //Historyfunktion gibt es nicht
350
351         //Emotionen neu setzen
352         player -> erhoeheNeutralHassGlobal(erhoehungGlobaleWut);
353         player -> setTrauerEuphorie ( freude );
354         player -> setSelbstsicherheitNervositaet ( nervositaet );
355
356
357 }
358
359 /**
360  * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: fold
361  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
362  * Ueberschreibt die Funktion aus EmoStrategy.
363  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
364  * \return Nothing
365  */
366 void DeterministischeEmoAnpassung::opponentFold(){
367         EmoPlayer* player = getPlayer ( );
368         player -> calcMyOdds();
369         double myOdds = player -> getMyOdds();
370
371         vector<int> alteWut = player -> getEmotion_hass();
372         int alteGlobaleWut = 0;
373         for(int i=0; i<alteWut.size(); i++){
374                 alteGlobaleWut = alteWut[i];
375         }
376         alteGlobaleWut = alteGlobaleWut / alteWut.size();
377
378         int alteFreude = player -> getEmotion_trauerEuphorie();
379         int alteAngst = player -> getEmotion_selbstsicherheitPanik();
380         int gegnerEinschaetzen = player -> getCharacter_gegnerEinschaetzen
                ();
381         int neueGlobaleWut = alteGlobaleWut;
382         int neueFreude = alteFreude;
383         int neueAngst = alteAngst;
384
385         switch(player -> getCurrentRound()){
386                 case 0:// vor dem Flop
387                         if (myOdds < 30){
388                                 neueGlobaleWut  = alteGlobaleWut * 0.99 +
                                        (60 - gegnerEinschaetzen) * 0.05;
389                                 neueFreude      = alteFreude * 1.01 + (
                                        gegnerEinschaetzen - 80) * 0.03;
390                                 neueAngst       = (int) alteAngst * 0.99;
391                         }else if (myOdds > 30 && myOdds < 70){
392                                 neueGlobaleWut  = alteGlobaleWut + (60 -
                                        gegnerEinschaetzen) * 0.06;
393                                 neueFreude      = alteFreude * 1.01 + (
                                        gegnerEinschaetzen - 82) * 0.03;
```

```
394                     }else if (myOdds > 70){
395                             neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                    (60 - gegnerEinschaetzen) * 0.06;
396                             neueFreude      = alteFreude * 0.99;
397                             neueAngst       = alteAngst * 1.01;
398                     }
399             break;
400
401             case 1: // nach dem Flop
402                     if (myOdds < 30){
403                             neueGlobaleWut  = alteGlobaleWut * 0.99 +
                                    (60 - gegnerEinschaetzen) * 0.06;
404                             neueFreude      = alteFreude * 1.01 + (
                                    gegnerEinschaetzen - 80) * 0.04;
405                             neueAngst       = alteAngst * 0.99;
406                     }else if (myOdds > 30 && myOdds < 70){
407                             neueGlobaleWut  = alteGlobaleWut + (55 -
                                    gegnerEinschaetzen) * 0.05;
408                             neueFreude      = alteFreude * 1.01 + (
                                    gegnerEinschaetzen - 84) * 0.03;
409                     }else if (myOdds > 70){
410                             neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                    (60 - gegnerEinschaetzen) * 0.06;
411                             neueFreude      = alteFreude * 0.98;
412                             neueAngst       = alteAngst * 1.02;
413                     }
414             break;
415
416             case 2: // nach dem Turn
417                     if (myOdds < 30){
418                             neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                    (60 - gegnerEinschaetzen) * 0.07;
419                             neueFreude      = alteFreude * 1.01 + (
                                    gegnerEinschaetzen - 80) * 0.05;
420                             neueAngst       = alteAngst * 0.98;
421                     }else if (myOdds > 30 && myOdds < 70){
422                             neueGlobaleWut  = alteGlobaleWut + (55 -
                                    gegnerEinschaetzen) * 0.04;
423                             neueFreude      = alteFreude * 1.01 + (
                                    gegnerEinschaetzen - 86) * 0.03;
424                     }else if (myOdds > 70){
425                             neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                    (60 - gegnerEinschaetzen) * 0.07;
426                             neueFreude      = alteFreude * 0.97;
427                             neueAngst       = alteAngst * 1.03;
428                     }
429             break;
430
431             case 3: // nach dem River
432                     if (myOdds < 30){
433                             neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                    (60 - gegnerEinschaetzen) * 0.07;
434                             neueFreude      = alteFreude * 1.02 + (
                                    gegnerEinschaetzen - 80) * 0.04;
435                             neueAngst       = alteAngst * 0.98;
436                     }else if (myOdds > 30 && myOdds < 70){
437                             neueGlobaleWut  = alteGlobaleWut + (55 -
                                    gegnerEinschaetzen) * 0.05;
438                             neueFreude      = alteFreude * 1.02 + (
                                    gegnerEinschaetzen - 86) * 0.05;
439                     }else if (myOdds > 70){
440                             neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                    (60 - gegnerEinschaetzen) * 0.07;
441                             neueFreude      = alteFreude * 0.99;
442                             neueAngst       = alteAngst * 1.04;
443                     }
444             break;
445
446             default: // <- sollte nicht eintreten
```

```
447                         // ggf. Fehlermeldung ausgeben
448                 break;
449         }
450
451         player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                alteGlobaleWut);
452         player -> setTrauerEuphorie((int)neueFreude);
453         player -> setSelbstsicherheitNervositaet((int)neueAngst);
454 }
455
456 /**
457  * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: check
458  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
459  * Ueberschreibt die Funktion aus EmoStrategy.
460  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
       den Charakterwerten des Spieler und den Karten abhaengen.
461  * \return Nothing
462  */
463 void DeterministischeEmoAnpassung::opponentCheck(){
464         EmoPlayer* player = getPlayer ( );
465         player -> calcMyOdds();
466         double myOdds = player -> getMyOdds();
467
468         vector<int> alteWut = player -> getEmotion_hass();
469         int alteGlobaleWut = 0;
470         for(int i=0; i<alteWut.size(); i++){
471                 alteGlobaleWut = alteWut[i];
472         }
473         alteGlobaleWut = alteGlobaleWut / alteWut.size();
474
475         int alteFreude = player -> getEmotion_trauerEuphorie();
476         int alteAngst = player -> getEmotion_selbstsicherheitPanik();
477         int gegnerEinschaetzen = player -> getCharacter_gegnerEinschaetzen
                ();
478         int neueGlobaleWut = alteGlobaleWut;
479         int neueFreude = alteFreude;
480         int neueAngst = alteAngst;
481
482         switch(player -> getCurrentRound()){
483                 case 0:// vor dem Flop
484                         if (myOdds < 30){
485                                 neueAngst       = alteAngst * 0.99;
486                         }else if (myOdds > 30 && myOdds < 70){
487                                 neueGlobaleWut  = alteGlobaleWut * 1.01;
488                                 neueFreude      = alteFreude * 0.99;
489                         }else if (myOdds > 70){
490                                 neueGlobaleWut  = alteGlobaleWut * 0.99;
491                                 neueFreude      = alteFreude * 1.02;
492                                 neueAngst       = alteAngst * 0.98;
493                         }
494                 break;
495
496                 case 1: // nach dem Flop
497                         if (myOdds < 30){
498                                 neueAngst       = alteAngst * 0.99;
499                         }else if (myOdds > 30 && myOdds < 70){
500                                 neueGlobaleWut  = alteGlobaleWut * 1.01;
501                                 neueFreude      = alteFreude * 0.99;
502                         }else if (myOdds > 70){
503                                 neueGlobaleWut  = alteGlobaleWut * 0.99;
504                                 neueFreude      = alteFreude * 1.02;
505                                 neueAngst       = alteAngst * 0.98;
506                         }
507                 break;
508
509                 case 2: // nach dem Turn
510                         if (myOdds < 30){
511                                 neueAngst       = alteAngst * 0.99;
512                         }else if (myOdds > 30 && myOdds < 70){
```

```
513                                     neueGlobaleWut  = alteGlobaleWut * 1.01;
514                                     neueFreude      = alteFreude * 0.99;
515                             }else if (myOdds > 70){
516                                     neueGlobaleWut  = alteGlobaleWut * 0.98;
517                                     neueFreude      = alteFreude * 1.03;
518                                     neueAngst       = alteAngst * 0.97;
519                             }
520                     break;
521
522                     case 3: // nach dem River
523                             if (myOdds < 30){
524                                     neueAngst       = alteAngst * 0.99;
525                             }else if (myOdds > 30 && myOdds < 70){
526                                     neueGlobaleWut  = alteGlobaleWut * 1.01;
527                                     neueFreude      = alteFreude * 0.99;
528                             }else if (myOdds > 70){
529                                     neueGlobaleWut  = alteGlobaleWut * 0.97;
530                                     neueFreude      = alteFreude * 1.05;
531                                     neueAngst       = alteAngst * 0.96;
532                             }
533                     break;
534
535                     default: // <- sollte nicht eintreten
536                             // ggf. Fehlermeldung ausgeben
537                     break;
538             }
539
540             player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                        alteGlobaleWut);
541             player -> setTrauerEuphorie((int)neueFreude);
542             player -> setSelbstsicherheitNervositaet((int)neueAngst);
543     }
544
545     /**
546      * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: raise
547      * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
548      * Ueberschreibt die Funktion aus EmoStrategy.
549      * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
                den Charakterwerten des Spieler und den Karten abhaengen.
550      * \return Nothing
551      */
552     void DeterministischeEmoAnpassung::opponentRaise(){
553             EmoPlayer* player = getPlayer ( );
554             player -> calcMyOdds();
555             double myOdds = player -> getMyOdds();
556
557             vector<int> alteWut = player -> getEmotion_hass();
558             int alteGlobaleWut = 0;
559             for(int i=0; i<alteWut.size(); i++){
560                     alteGlobaleWut = alteWut[i];
561             }
562             alteGlobaleWut = alteGlobaleWut / alteWut.size();
563
564             int alteFreude = player -> getEmotion_trauerEuphorie();
565             int alteAngst = player -> getEmotion_selbstsicherheitPanik();
566             int gegnerEinschaetzen = player -> getCharacter_gegnerEinschaetzen
                        ();
567             int anpassungsfaehigkeit = player ->
                        getCharacter_anpassungsfaehigkeit();
568             int neueGlobaleWut = alteGlobaleWut;
569             int neueFreude = alteFreude;
570             int neueAngst = alteAngst;
571
572             switch(player -> getCurrentRound()){
573                     case 0:// vor dem Flop
574                             if (myOdds < 30){
575                                     neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                            (75 - gegnerEinschaetzen) * 0.1 + (50
                                            - anpassungsfaehigkeit) *0.05;
```

```
576                                 neueFreude      = alteFreude * 0.98 + (
                                        gegnerEinschaetzen - 80) * 0.1;
577                                 neueAngst       = alteAngst * 1.02 + (75 -
                                         gegnerEinschaetzen) * 0.1 + (50 -
                                        anpassungsfaehigkeit) * 0.05;
578                         }else if (myOdds > 30 && myOdds < 70){
579                                 neueGlobaleWut  = alteGlobaleWut + (75 -
                                        gegnerEinschaetzen) * 0.1;
580                                 neueAngst       = alteAngst * 1.01 + (75 -
                                         gegnerEinschaetzen) * 0.05 + (50 -
                                        anpassungsfaehigkeit) * 0.03;
581                         }else if (myOdds > 70){
582                                 neueGlobaleWut  = alteGlobaleWut * 0.97 +
                                        (50 - gegnerEinschaetzen) * 0.1;
583                                 neueFreude      = alteFreude * 1.03 + (
                                        gegnerEinschaetzen - 65) * 0.1;
584                                 neueAngst       = alteAngst * 0.99 + (
                                        gegnerEinschaetzen - 50) * 0.1;
585                         }
586                 break;
587
588                 case 1: // nach dem Flop
589                         if (myOdds < 30){
590                                 neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                        (80 - gegnerEinschaetzen) * 0.1 + (55
                                        - anpassungsfaehigkeit) * 0.05;
591                                 neueFreude      = alteFreude * 0.97 + (
                                        gegnerEinschaetzen - 85) * 0.1;
592                                 neueAngst       = alteAngst * 1.03 + (80 -
                                         gegnerEinschaetzen) * 0.1 + (50 -
                                        anpassungsfaehigkeit) * 0.03;
593                         }else if (myOdds > 30 && myOdds < 70){
594                                 neueGlobaleWut  = alteGlobaleWut + (75 -
                                        gegnerEinschaetzen) * 0.1;
595                                 neueAngst       = alteAngst * 1.01 + (75 -
                                         gegnerEinschaetzen) * 0.03;
596                         }else if (myOdds > 70){
597                                 neueGlobaleWut  = alteGlobaleWut * 0.96 +
                                        (50 - gegnerEinschaetzen) * 0.1;
598                                 neueFreude      = alteFreude * 1.04 + (
                                        gegnerEinschaetzen - 60) * 0.1;
599                                 neueAngst       = alteAngst * 0.99 + (
                                        gegnerEinschaetzen - 50) * 0.1;
600                         }
601                 break;
602
603                 case 2: // nach dem Turn
604                         if (myOdds < 30){
605                                 neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                        (85 - gegnerEinschaetzen) * 0.1 + (60
                                        - anpassungsfaehigkeit) * 0.05;
606                                 neueFreude      = alteFreude * 0.96 + (
                                        gegnerEinschaetzen - 88) * 0.1;
607                                 neueAngst       = alteAngst * 1.04 + (85 -
                                         gegnerEinschaetzen) * 0.1 + (50 -
                                        anpassungsfaehigkeit) * 0.02;
608                         }else if (myOdds > 30 && myOdds < 70){
609                                 neueGlobaleWut  = alteGlobaleWut + (75 -
                                        gegnerEinschaetzen) * 0.1;
610                                 neueAngst       = alteAngst * 1.01 + (75 -
                                         gegnerEinschaetzen) * 0.02;
611                         }else if (myOdds > 70){
612                                 neueGlobaleWut  = alteGlobaleWut * 0.95 +
                                        (50 - gegnerEinschaetzen) * 0.1;
613                                 neueFreude      = alteFreude * 1.05 + (
                                        gegnerEinschaetzen - 60) * 0.1;
614                                 neueAngst       = alteAngst * 0.99 + (
                                        gegnerEinschaetzen - 50) * 0.1;
615                         }
```

```
616                 break;
617
618                 case 3: // nach dem River
619                         if (myOdds < 30){
620                                 neueGlobaleWut  = alteGlobaleWut * 1.05 +
                                        (88 - gegnerEinschaetzen) * 0.1 + (63
                                        - anpassungsfaehigkeit) * 0.05;
621                                 neueFreude      = alteFreude * 0.95 + (
                                        gegnerEinschaetzen - 89) * 0.1;
622                                 neueAngst       = alteAngst * 1.05 + (87 -
                                         gegnerEinschaetzen) * 0.1 + (50 -
                                        anpassungsfaehigkeit) * 0.01;
623                         }else if (myOdds > 30 && myOdds < 70){
624                                 neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                        (75 - gegnerEinschaetzen) * 0.1;
625                                 neueFreude      = alteFreude * 0.98 + (
                                        gegnerEinschaetzen - 85) * 0.05;
626                                 neueAngst       = alteAngst * 1.01 + (75 -
                                         gegnerEinschaetzen) * 0.02;
627                         }else if (myOdds > 70){
628                                 neueGlobaleWut  = alteGlobaleWut * 0.94 +
                                        (50 - gegnerEinschaetzen) * 0.1;
629                                 neueFreude      = alteFreude * 1.05 + (
                                        gegnerEinschaetzen - 60) * 0.1;
630                                 neueAngst       = alteAngst * 0.98 + (
                                        gegnerEinschaetzen - 50) * 0.1;
631                         }
632                 break;
633
634                 default: // <- sollte nicht eintreten
635                         // ggf. Fehlermeldung ausgeben
636                 break;
637         }
638
639         player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                alteGlobaleWut);
640         player -> setTrauerEuphorie((int)neueFreude);
641         player -> setSelbstsicherheitNervositaet((int)neueAngst);
642 }
643
644 /**
645  * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: call
646  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
647  * Ueberschreibt die Funktion aus EmoStrategy.
648  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
649  * \return Nothing
650  */
651 void DeterministischeEmoAnpassung::opponentCall(){
652         EmoPlayer* player = getPlayer ( );
653         player -> calcMyOdds();
654         double myOdds = player -> getMyOdds();
655
656         vector<int> alteWut = player -> getEmotion_hass();
657         int alteGlobaleWut = 0;
658         for(int i=0; i<alteWut.size(); i++){
659                 alteGlobaleWut = alteWut[i];
660         }
661         alteGlobaleWut = alteGlobaleWut / alteWut.size();
662
663         int alteFreude = player -> getEmotion_trauerEuphorie();
664         int alteAngst = player -> getEmotion_selbstsicherheitPanik();
665         int gegnerEinschaetzen = player -> getCharacter_gegnerEinschaetzen
                ();
666         int gedaechtnis = player ->  getCharacter_gedaechtnis();
667         int neueGlobaleWut = alteGlobaleWut;
668         int neueFreude = alteFreude;
669         int neueAngst = alteAngst;
670
```

```
671         switch(player -> getCurrentRound()){
672                 case 0:// vor dem Flop
673                         if (myOdds < 30){
674                                 // keine Aenderungen
675                         }else if (myOdds > 30 && myOdds < 70){
676                                 neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                        (85 - gedaechtnis) * 0.03;
677                                 neueFreude      = alteFreude + (80 -
                                        gedaechtnis) * 0.02 + (70 -
                                        gegnerEinschaetzen) * 0.02;
678                         }else if (myOdds > 70){
679                                 neueGlobaleWut  = alteGlobaleWut * 0.99 +
                                        (80 - gedaechtnis) * 0.2;
680                                 neueFreude      = alteFreude * 1.01 + (
                                        gedaechtnis - 75) * 0.1 + (
                                        gegnerEinschaetzen - 80) * 0.02;
681                                 neueAngst       = alteAngst * 0.99 -
                                        gegnerEinschaetzen * 0.02;
682                         }
683                 break;
684
685                 case 1: // nach dem Flop
686                         if (myOdds < 30){
687                                 neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                        (87 - gedaechtnis) * 0.2;
688                                 neueFreude      = alteFreude * 0.99 + (
                                        gegnerEinschaetzen - 75) * 0.04;
689                                 neueAngst       = alteAngst * 1.01;
690                         }else if (myOdds > 30 && myOdds < 70){
691                                 neueGlobaleWut  = alteGlobaleWut + (85 -
                                        gedaechtnis) * 0.03;
692                                 neueFreude      = alteFreude + (82 -
                                        gedaechtnis) * 0.02 + (67 -
                                        gegnerEinschaetzen) * 0.02;
693                         }else if (myOdds > 70){
694                                 neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                        (71 - gedaechtnis) * 0.2;
695                                 neueFreude      = alteFreude * 1.02 + (
                                        gedaechtnis - 75) * 0.2 + (
                                        gegnerEinschaetzen - 74) * 0.03;
696                                 neueAngst       = alteAngst -
                                        gegnerEinschaetzen * 0.02;
697                         }
698                 break;
699
700                 case 2: // nach dem Turn
701                         if (myOdds < 30){
702                                 neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                        (90 - gedaechtnis) * 0.2;
703                                 neueFreude      = alteFreude * 0.98 + (
                                        gegnerEinschaetzen - 77) * 0.04;
704                                 neueAngst       = alteAngst * 1.01 + (
                                        gegnerEinschaetzen - 67) * 0.02;
705                         }else if (myOdds > 30 && myOdds < 70){
706                                 neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                        (85 - gedaechtnis) * 0.02;
707                                 neueFreude      = alteFreude + (85 -
                                        gedaechtnis) * 0.02 + (64 -
                                        gegnerEinschaetzen) * 0.02;
708                         }else if (myOdds > 70){
709                                 neueGlobaleWut  = alteGlobaleWut * 0.97 +
                                        (70 - gedaechtnis) * 0.2;
710                                 neueFreude      = alteFreude * 1.03 + (
                                        gedaechtnis - 75) * 0.2 + (
                                        gegnerEinschaetzen - 71) * 0.03;
711                                 neueAngst       = alteAngst + (80 -
                                        gegnerEinschaetzen) * 0.03;
712                         }
713                 break;
```

```
714
715                 case 3: // nach dem River
716                         if (myOdds < 30){
717                                 neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                        (92 - gedaechtnis) * 0.1;
718                                 neueFreude      = alteFreude * 0.97 + (
                                        gegnerEinschaetzen - 78) * 0.03;
719                                 neueAngst       = alteAngst * 1.02 + (
                                        gegnerEinschaetzen - 69) * 0.01;
720                         }else if (myOdds > 30 && myOdds < 70){
721                                 neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                        (87 - gedaechtnis) * 0.02;
722                                 neueFreude      = alteFreude + (86 -
                                        gedaechtnis) * 0.02 + (65 -
                                        gegnerEinschaetzen) * 0.02;
723                                 neueAngst       = alteAngst * 1.02;
724                         }else if (myOdds > 70){
725                                 neueGlobaleWut  = alteGlobaleWut * 0.97 +
                                        (gedaechtnis - 65) * 0.2;
726                                 neueFreude      = alteFreude * 1.03 + (
                                        gedaechtnis - 74) * 0.2 + (
                                        gegnerEinschaetzen - 68) * 0.02;
727                                 neueAngst       = alteAngst * 1.03 + (75 -
                                         gegnerEinschaetzen) * 0.03;
728                         }
729                 break;
730
731                 default: // <- sollte nicht eintreten
732                         // ggf. Fehlermeldung ausgeben
733                 break;
734         }
735
736         player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                alteGlobaleWut);
737         player -> setTrauerEuphorie((int)neueFreude);
738         player -> setSelbstsicherheitNervositaet((int)neueAngst);
739 }
740
741 /**
742  * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: bet
743  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
744  * Ueberschreibt die Funktion aus EmoStrategy.
745  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
746  * \return Nothing
747  */
748 void DeterministischeEmoAnpassung::opponentBet(){
749         EmoPlayer* player = getPlayer ( );
750         player -> calcMyOdds();
751         double myOdds = player -> getMyOdds();
752
753         vector<int> alteWut = player -> getEmotion_hass();
754         int alteGlobaleWut = 0;
755         for(int i=0; i<alteWut.size(); i++){
756                 alteGlobaleWut = alteWut[i];
757         }
758         alteGlobaleWut = alteGlobaleWut / alteWut.size();
759
760         int alteFreude = player -> getEmotion_trauerEuphorie();
761         int alteAngst = player -> getEmotion_selbstsicherheitPanik();
762         int gegnerEinschaetzen = player -> getCharacter_gegnerEinschaetzen
                ();
763         int anpassungsfaehigkeit = player ->
                getCharacter_anpassungsfaehigkeit();
764         int neueGlobaleWut = alteGlobaleWut;
765         int neueFreude = alteFreude;
766         int neueAngst = alteAngst;
767
768         switch(player -> getCurrentRound()){
```

```
769                     case 0:// vor dem Flop
770                             if (myOdds < 30){
771                                     neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                            (75 - gegnerEinschaetzen) * 0.1 + (50
                                            - anpassungsfaehigkeit) *0.05;
772                                     neueFreude      = alteFreude * 0.99 + (
                                            gegnerEinschaetzen - 80) * 0.1;
773                                     neueAngst       = alteAngst * 1.01 + (75 -
                                             gegnerEinschaetzen) * 0.1 + (50 -
                                            anpassungsfaehigkeit) * 0.05;
774                             }else if (myOdds > 30 && myOdds < 70){
775                                     neueGlobaleWut  = alteGlobaleWut + (75 -
                                            gegnerEinschaetzen) * 0.1;
776                                     neueAngst       = alteAngst * 1.01 + (75 -
                                             gegnerEinschaetzen) * 0.05 + (50 -
                                            anpassungsfaehigkeit) * 0.03;
777                             }else if (myOdds > 70){
778                                     neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                            (50 - gegnerEinschaetzen) * 0.1;
779                                     neueFreude      = alteFreude * 1.02 + (
                                            gegnerEinschaetzen - 65) * 0.1;
780                                     neueAngst       = alteAngst * 0.99 + (
                                            gegnerEinschaetzen - 50) * 0.1;
781                             }
782                     break;
783
784                     case 1: // nach dem Flop
785                             if (myOdds < 30){
786                                     neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                            (80 - gegnerEinschaetzen) * 0.1 + (55
                                            - anpassungsfaehigkeit) * 0.05;
787                                     neueFreude      = alteFreude * 0.98 + (
                                            gegnerEinschaetzen - 85) * 0.1;
788                                     neueAngst       = alteAngst * 1.02 + (80 -
                                             gegnerEinschaetzen) * 0.1 + (50 -
                                            anpassungsfaehigkeit) * 0.03;
789                             }else if (myOdds > 30 && myOdds < 70){
790                                     neueGlobaleWut  = alteGlobaleWut + (75 -
                                            gegnerEinschaetzen) * 0.1;
791                                     neueAngst       = alteAngst * 1.01 + (75 -
                                             gegnerEinschaetzen) * 0.03;
792                             }else if (myOdds > 70){
793                                     neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                            (50 - gegnerEinschaetzen) * 0.1;
794                                     neueFreude      = alteFreude * 1.02 + (
                                            gegnerEinschaetzen - 60) * 0.1;
795                                     neueAngst       = alteAngst * 0.99 + (
                                            gegnerEinschaetzen - 50) * 0.1;
796                             }
797                     break;
798
799                     case 2: // nach dem Turn
800                             if (myOdds < 30){
801                                     neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                            (85 - gegnerEinschaetzen) * 0.1 + (60
                                            - anpassungsfaehigkeit) * 0.05;
802                                     neueFreude      = alteFreude * 0.98 + (
                                            gegnerEinschaetzen - 88) * 0.1;
803                                     neueAngst       = alteAngst * 1.02 + (85 -
                                             gegnerEinschaetzen) * 0.1 + (50 -
                                            anpassungsfaehigkeit) * 0.02;
804                             }else if (myOdds > 30 && myOdds < 70){
805                                     neueGlobaleWut  = alteGlobaleWut + (75 -
                                            gegnerEinschaetzen) * 0.1;
806                                     neueAngst       = alteAngst * 1.01 + (75 -
                                             gegnerEinschaetzen) * 0.02;
807                             }else if (myOdds > 70){
808                                     neueGlobaleWut  = alteGlobaleWut * 0.97 +
                                            (50 - gegnerEinschaetzen) * 0.1;
```

```
809                                     neueFreude      = alteFreude * 1.02 + (
                                            gegnerEinschaetzen - 60) * 0.1;
810                                     neueAngst       = alteAngst * 0.99 + (
                                            gegnerEinschaetzen - 50) * 0.1;
811                             }
812                     break;
813
814                     case 3: // nach dem River
815                             if (myOdds < 30){
816                                     neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                            (88 - gegnerEinschaetzen) * 0.1 + (63
                                            - anpassungsfaehigkeit) * 0.05;
817                                     neueFreude      = alteFreude * 0.97 + (
                                            gegnerEinschaetzen - 89) * 0.1;
818                                     neueAngst       = alteAngst * 1.03 + (87 -
                                             gegnerEinschaetzen) * 0.1 + (50 -
                                            anpassungsfaehigkeit) * 0.01;
819                             }else if (myOdds > 30 && myOdds < 70){
820                                     neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                            (75 - gegnerEinschaetzen) * 0.1;
821                                     neueFreude      = alteFreude * 0.99 + (
                                            gegnerEinschaetzen - 85) * 0.05;
822                                     neueAngst       = alteAngst * 1.01 + (75 -
                                             gegnerEinschaetzen) * 0.02;
823                             }else if (myOdds > 70){
824                                     neueGlobaleWut  = alteGlobaleWut * 0.96 +
                                            (50 - gegnerEinschaetzen) * 0.1;
825                                     neueFreude      = alteFreude * 1.03 + (
                                            gegnerEinschaetzen - 60) * 0.1;
826                                     neueAngst       = alteAngst * 0.99 + (
                                            gegnerEinschaetzen - 50) * 0.1;
827                             }
828                     break;
829
830                     default: // <- sollte nicht eintreten
831                             // ggf. Fehlermeldung ausgeben
832                     break;
833             }
834
835             player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                        alteGlobaleWut);
836             player -> setTrauerEuphorie((int)neueFreude);
837             player -> setSelbstsicherheitNervositaet((int)neueAngst);
838     }
839
840     /**
841      * Veraendert die Emotionen des Spieler m_player bei Gegneraktion: All-In
842      * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
843      * Ueberschreibt die Funktion aus EmoStrategy.
844      * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
                den Charakterwerten des Spieler und den Karten abhaengen.
845      * \return Nothing
846      */
847     void DeterministischeEmoAnpassung::opponentAllIn(){
848             EmoPlayer* player = getPlayer ( );
849             player -> calcMyOdds();
850             double myOdds = player -> getMyOdds();
851
852             vector<int> alteWut = player -> getEmotion_hass();
853             int alteGlobaleWut = 0;
854             for(int i=0; i<alteWut.size(); i++){
855                     alteGlobaleWut = alteWut[i];
856             }
857             alteGlobaleWut  = alteGlobaleWut / alteWut.size();
858
859             int alteFreude  = player -> getEmotion_trauerEuphorie();
860             int alteAngst   = player -> getEmotion_selbstsicherheitPanik();
861
```

```
862         int gegnerEinschaetzen  = player ->
                getCharacter_gegnerEinschaetzen();
863         int setzverhaltenGeld   = player -> getCharacter_setzverhaltenGeld
                ();
864         int spielverhaltenHand  = player -> getCharacter_setzverhaltenHand
                ();
865         int gedaechtnis         = player -> getCharacter_gedaechtnis();
866
867         int neueGlobaleWut      = alteGlobaleWut;
868         int neueFreude          = alteFreude;
869         int neueAngst           = alteAngst;
870
871         switch(player -> getCurrentRound()){
872                 case 0:// vor dem Flop
873                         if (myOdds < 30){
874                                 neueFreude      = alteFreude * 0.99;
875                         }else if (myOdds > 30 && myOdds < 70){
876                                 neueGlobaleWut  = alteGlobaleWut * 1.01 +
                                        (75 - gegnerEinschaetzen) * 0.01 + (
                                        spielverhaltenHand - 40) * 0.02 + (50
                                        - gedaechtnis) * 0.01;
877                                 neueFreude      = alteFreude + (
                                        gedaechtnis - 50) * 0.01 + (
                                        gegnerEinschaetzen - 75) * 0.01 + (
                                        spielverhaltenHand - 40) * 0.02;
878                                 neueAngst       = alteAngst + (
                                        setzverhaltenGeld - 50) * 0.01 - (
                                        gedaechtnis - 50) * 0.01;
879
880                         }else if (myOdds > 70){
881                                 neueGlobaleWut  = alteGlobaleWut + (60 -
                                        gegnerEinschaetzen) * 0.01 + (50 -
                                        spielverhaltenHand) * 0.01 + (50 -
                                        gedaechtnis) * 0.01;
882                                 neueFreude      = alteFreude + (
                                        gedaechtnis - 50) * 0.01 + (
                                        gegnerEinschaetzen - 75) * 0.01 + (
                                        spielverhaltenHand - 40) * 0.02;
883                                 neueAngst       = alteAngst + (
                                        setzverhaltenGeld - 50) * 0.01 - (
                                        gedaechtnis - 50) * 0.01;
884                         }
885                 break;
886
887                 case 1: // nach dem Flop
888                         if (myOdds < 30){
889                                 neueFreude      = alteFreude * 0.99 + (
                                        gegnerEinschaetzen - 75) * 0.01;
890                                 neueGlobaleWut  = alteGlobaleWut * 1.01;
891                         }else if (myOdds > 30 && myOdds < 70){
892                                 neueGlobaleWut  = alteGlobaleWut * 1.02 +
                                        (74 - gegnerEinschaetzen) * 0.02+ (
                                        spielverhaltenHand - 41) * 0.03 + (50
                                        - gedaechtnis) * 0.01;
893                                 neueFreude      = alteFreude + (
                                        gedaechtnis - 52) * 0.02 + (
                                        gegnerEinschaetzen - 75) * 0.02 + (
                                        spielverhaltenHand - 40) * 0.02;
894                                 neueAngst       = alteAngst + (
                                        setzverhaltenGeld - 53) * 0.01 - (
                                        gedaechtnis - 50) * 0.02;
895
896                         }else if (myOdds > 70){
897                                 neueGlobaleWut  = alteGlobaleWut + (61 -
                                        gegnerEinschaetzen) * 0.02 + (50 -
                                        spielverhaltenHand) * 0.02 + (50 -
                                        gedaechtnis) * 0.01;
898                                 neueFreude      = alteFreude * 1.01 + (
                                        gedaechtnis - 50) * 0.02 + (
```

```
                                    gegnerEinschaetzen - 75) * 0.01 + (
                                    spielverhaltenHand - 40) * 0.02;
899                             neueAngst       = alteAngst + (
                                    setzverhaltenGeld - 50) * 0.02 - (
                                    gedaechtnis - 50) * 0.01;
900                     }
901             break;
902
903             case 2: // nach dem Turn
904                     if (myOdds < 30){
905                             neueFreude      = alteFreude * 0.98 + (
                                    gegnerEinschaetzen - 75) * 0.015;
906                             neueGlobaleWut  = alteGlobaleWut * 1.02;
907                     }else if (myOdds > 30 && myOdds < 70){
908                             neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                    (74 - gegnerEinschaetzen) * 0.03 + (
                                    spielverhaltenHand - 44) * 0.03 + (50
                                    - gedaechtnis) * 0.01;
909                             neueFreude      = alteFreude + (
                                    gedaechtnis - 52) * 0.03 + (
                                    gegnerEinschaetzen - 75) * 0.02 + (
                                    spielverhaltenHand - 40) * 0.02;
910                             neueAngst       = alteAngst + (
                                    setzverhaltenGeld - 53) * 0.02 - (
                                    gedaechtnis - 50) * 0.03;
911
912                     }else if (myOdds > 70){
913                             neueGlobaleWut  = alteGlobaleWut * 0.99 +
                                    (61 - gegnerEinschaetzen) * 0.02 + (50
                                     - spielverhaltenHand) * 0.02 + (50 -
                                    gedaechtnis) * 0.01;
914                             neueFreude      = alteFreude * 1.01 + (
                                    gedaechtnis - 50) * 0.02 + (
                                    gegnerEinschaetzen - 75) * 0.01 + (
                                    spielverhaltenHand - 40) * 0.03;
915                             neueAngst       = alteAngst + (
                                    setzverhaltenGeld - 50) * 0.02 - (
                                    gedaechtnis - 50) * 0.01;
916                     }
917             break;
918
919             case 3: // nach dem River
920                     if (myOdds < 30){
921                             neueFreude      = alteFreude * 0.97 + (
                                    gegnerEinschaetzen - 75) * 0.015;
922                             neueGlobaleWut  = alteGlobaleWut * 1.03;
923                     }else if (myOdds > 30 && myOdds < 70){
924                             neueGlobaleWut  = alteGlobaleWut * 1.03 +
                                    (74 - gegnerEinschaetzen) * 0.04 + (
                                    spielverhaltenHand - 44) * 0.04 + (50
                                    - gedaechtnis) * 0.02;
925                             neueFreude      = alteFreude + (
                                    gedaechtnis - 52) * 0.04 + (
                                    gegnerEinschaetzen - 75) * 0.03 + (
                                    spielverhaltenHand - 40) * 0.02;
926                             neueAngst       = alteAngst + (
                                    setzverhaltenGeld - 53) * 0.03 - (
                                    gedaechtnis - 50) * 0.04;
927
928                     }else if (myOdds > 70){
929                             neueGlobaleWut  = alteGlobaleWut * 0.98 +
                                    (61 - gegnerEinschaetzen) * 0.02 + (50
                                     - spielverhaltenHand) * 0.02 + (50 -
                                    gedaechtnis) * 0.01;
930                             neueFreude      = alteFreude * 1.02 + (
                                    gedaechtnis - 50) * 0.02 + (
                                    gegnerEinschaetzen - 75) * 0.01 + (
                                    spielverhaltenHand - 40) * 0.03;
```

```
931                                     neueAngst       = alteAngst * 1.01+ (
                                            setzverhaltenGeld - 50) * 0.02 - (
                                            gedaechtnis - 50) * 0.02;
932                             }
933                     break;
934
935                     default: // <- sollte nicht eintreten
936                             // ggf. Fehlermeldung ausgeben
937                     break;
938             }
939
940             player -> erhoeheNeutralHassGlobal((int)neueGlobaleWut-
                    alteGlobaleWut);
941             player -> setTrauerEuphorie((int)neueFreude);
942             player -> setSelbstsicherheitNervositaet((int)neueAngst);
943     }
944
945
946     /**
947      * Ermittelt das Maximum des Geldes was, andere Spieler haben, die noch im
            Spiel sind
948      */
949     int DeterministischeEmoAnpassung::getmaxchips_of_others()
950     {
951      EmoPlayer* player = getPlayer();
952
953      int maxchips = 0;
954
955      PlayerListConstIterator it_c;
956      /*for(it_c=player->getHand()->getActivePlayerList()->begin();
957     it_c!=getHand()->getActivePlayerList()->end(); it_c++)
958       {
959        if((*it_c)->getMyCash()> maxchips && (*it_c)->getMyID() != player->
               getMyID()) // hat ein Spieler der NICHT 'ich' ist mehr geld als ich
                bisher weiss, merk ich mir den wert
960         {
961          maxchips = (*it_c)->getMyCash();
962         }
963       }*/
964      return maxchips;
965
966      //return 50;
967     }
968
969     /**
970      * gibt die Gewinnchancen des Spielers in seiner aktuellen Situation
            wieder. Eine normierung des Situationsbedingten
            Wahrscheinlichkeitsbereiches wird noch vorgenommen.
971      */
972     int DeterministischeEmoAnpassung::Kartentrans()
973     {
974       EmoPlayer* player = getPlayer();
975       player->calcMyOdds();
976       return (int)(player->getMyOdds());
977       //return 50;
978     }
979
980     /**
981      * normZZ erzeugt normalverteilte Zufallszahlen aus dem Intervall [0;1].
982      */
983     double DeterministischeEmoAnpassung::normZZ(){
984      double N_0_1 = 0;
985
986      srand((unsigned) time(NULL));
987
988      for(int i= 1; i<=12; i++)
989       {
990        N_0_1 +=(double)rand()/(double)2147483648;
991         //oder das nehmen
```

```
992       //[MatNish98]
993       //Makoto Matsumoto, Takuji Nischimura: Mersenne Twister: A 623-
              dimensionally
994       //equiditributed uniform pseudorandom number generator, Keio
              University, Yokohama
995       //1998
996     }
997   return (N_0_1 -6);  //zentraler Grenzwertsatz
998 }
999
1000 /**
1001  * Veraendert die Emotionen des Spieler m_player bei eigener Aktion: All-
        In
1002  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
1003  * Ueberschreibt die Funktion aus EmoStrategy.
1004  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
1005  */
1006 void DeterministischeEmoAnpassung::ownAllIn(unsigned int mySet)
1007 {
1008
1009   EmoPlayer* player = getPlayer();
1010   int delta_wut = 0;
1011   int delta_freude = 0;
1012   int delta_nervoes = 0;
1013
1014   switch(player->getMyAction())
1015     {
1016       case 3:  //call
1017                {
1018                 delta_nervoes = (int)(60 * (getmaxchips_of_others() / mySet)
                         * ((100-Kartentrans())/100 + normZZ()*(100- player->
                         getCharacter_kartenBewerten())) );
1019                 delta_wut     = (int)(80 * (getmaxchips_of_others() / mySet)
                         * ((100-Kartentrans())/100 + normZZ()*(100- player->
                         getCharacter_kartenBewerten())) );
1020                 delta_freude  =(int)(-60 * (getmaxchips_of_others() / mySet)
                         * ((100-Kartentrans())/100 + normZZ()*(100- player->
                         getCharacter_kartenBewerten())) );
1021                }
1022               break;
1023       case 4:  //bet
1024       case 5:  //raise
1025                {
1026                 delta_nervoes = (int)(70 * getmaxchips_of_others() / (mySet)
                         * ((100-Kartentrans())/100 + normZZ()*(100- player->
                         getCharacter_kartenBewerten())) );
1027                 delta_freude  = (int)(90 * mySet / getmaxchips_of_others()
                         * ((100-Kartentrans())/100 + normZZ()*(100- player->
                         getCharacter_kartenBewerten())) );
1028                }
1029               break;
1030       default: std::cout << "Fehler: ownAllIn wurde aufgerufen im Fall " <<
                player->getMyAction() << endl;
1031                break;
1032     }
1033
1034    player -> erhoeheNeutralHassGlobal(delta_wut);
1035    player->setTrauerEuphorie(player->getEmotion_trauerEuphorie()+
           delta_freude);
1036    player->setSelbstsicherheitNervositaet(player->
           getEmotion_selbstsicherheitPanik()+delta_nervoes);
1037
1038 }
1039
1040 /**
1041  * Veraendert die Emotionen des Spieler m_player bei eigenen Aktionen:
        call und raise/bet
1042  * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
```

```
1043  * Ueberschreibt die Funktion aus EmoStrategy.
1044  * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
        den Charakterwerten des Spieler und den Karten abhaengen.
1045  * \param einsatz        Der Betrag mit dem der Spieler raise oder call
        durchfuhrt
1046  */
1047 void DeterministischeEmoAnpassung::ownCallRaise(unsigned int einsatz, bool
      raise){
1048
1049   EmoPlayer* player = getPlayer();
1050
1051   int callraise = 0;
1052   if(raise)
1053    callraise = 1;
1054   //switch(player->getMyAction()){
1055   //   case 3:  //call
1056   //            break;
1057   //   case 4:  //bet
1058   //   case 5:  //raise
1059   //            callraise = 1;
1060   //            break;
1061   //   default: std::cout << "Fehler: ownCallRaise wurde aufgerufen im Fall
         "<< player->getMyAction() << endl;
1062   //             break;
1063   //}
1064   //einsatz = player-> getMySet();
1065
1066   //x = 0(call) 1(raise)
1067   //y = 0 (wenig gesetzt) 1 (mittel gesetzt) 2 (viel gesetzt)
1068   //z = 0(preflop); 1 (postflop); 2 (postturn); 3 (postriver)
1069   //double[x][y][z]
1070
1071   double weights[2][3][4];
1072   // [-4;6] * weights   abhaengig von bias
1073
1074   //call
1075   //wenig gesetzt
1076   weights[0][0][0] = 3; //[-12;18]
1077   weights[0][0][1] = 4; //[-16;24]
1078   weights[0][0][2] = 6; //[-24;36]
1079   weights[0][0][3] = 8; //[-32;48]
1080   //mittel gesetzt
1081   weights[0][1][0] = 6; //[-24;36]
1082   weights[0][1][1] = 7; //[-28;42]
1083   weights[0][1][2] = 8; //[-32;48]
1084   weights[0][0][3] = 9; //[-36;54]
1085   //viel gesetzt
1086   weights[0][2][0] = 10; //[-40;60]
1087   weights[0][2][1] = 11; //[-44;66]
1088   weights[0][2][2] = 12; //[-48;72]
1089   weights[0][2][3] = 13; //[-52;78]
1090
1091   //raise
1092   //wenig gesetzt
1093   weights[1][0][0] = 8; //[-32;48]
1094   weights[1][0][1] = 8.5; //[-34;51]
1095   weights[1][0][2] = 9.5; //[-38;57]
1096   weights[1][0][3] = 10; //[-40;60]
1097   //mittel gesetzt
1098   weights[1][1][0] = 8.5; //[-34;51]
1099   weights[1][1][1] = 9; //-36;54]
1100   weights[1][1][2] = 9.5; //[-38;57]
1101   weights[1][1][3] = 10; //[-40;60]
1102   //viel gesetzt
1103   weights[1][2][0] = 12; //[-48;72]
1104   weights[1][2][1] = 13; //[-52;78]
1105   weights[1][2][2] = 14.5; //[-58;87]
1106   weights[1][2][3] = 15; //[-60;90]
1107
```

```
1108    //bias schmaelert den bereich, den man nervoes wird, wenn man schlechte
            karten hat, da man sich fuer diese aktion bewusst entschieden hat
1109    //bias = 100 -> keie schmaelerung ; bias = 0 kein negativer Bereich fuer
            nervositaet
1110    int bias = 60;
1111
1112    int X = Kartentrans() - 50;
1113
1114    X = (X<0) ? (X*bias/100) : X;
1115
1116    double sigma = (100- player->getCharacter_kartenBewerten())/10;
1117
1118    double N_0_1 = normZZ();
1119
1120    // Delta = X + N(0; (10-blattW)/10) *10 =>
1121    // Delta = X + N_0_1* 10-blattW /10 *10 = X + N_0_1*(10-blatt)
1122
1123    int wenigMittelViel=0;
1124    if(einsatz<500)
1125     {wenigMittelViel = 0;}
1126    else if(einsatz<4000)
1127     {wenigMittelViel = 1;}
1128    else
1129     {wenigMittelViel = 2;}
1130
1131    X += (int)(N_0_1*sigma);
1132    X *= weights[callraise][wenigMittelViel][player->getCurrentRound()];
1133    player->setSelbstsicherheitNervositaet(player->
            getEmotion_selbstsicherheitPanik()+X);
1134
1135  }
1136
1137  /**
1138   * Veraendert die Emotionen des Spieler m_player bei eigener Aktion: check
1139   * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
1140   * Ueberschreibt die Funktion aus EmoStrategy.
1141   * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
           den Charakterwerten des Spieler und den Karten abhaengen.
1142   */
1143  void DeterministischeEmoAnpassung::ownCheck()
1144  {
1145
1146
1147    EmoPlayer* player = getPlayer();
1148
1149    //x = 0(preflop); 1 (postflop); 2 (postturn); 3 (postriver)
1150    //double[x]
1151    double weights[4];
1152    // [-6;6] * weights   abhaengig von bias
1153    //Spielphasen
1154    weights[0] = 3; //[-18;18]
1155    weights[1] = 4; //[-24;24]
1156    weights[2] = 6; //[-36;36]
1157    weights[3] = 8; //[-48;48]
1158
1159    //bias schmaelert den bereich, den man nervoes ist, wenn man schlechte
            karten hat, da man sich fuer diese aktion bewusst entschieden hat
1160    //bias = 100 -> keie schmaelerung ; bias = 0 kein negativer Bereich fuer
            nervositaet
1161    int bias = 60;
1162
1163    int X = Kartentrans() - 50;
1164    X = (X<0) ? (X*bias/100) : X;
1165
1166    double sigma = (100- player->getCharacter_kartenBewerten())/30;
1167
1168    double N_0_1 = normZZ();
1169
1170    // Delta = X + N(0; (10-blattW)/10) *10   =>
```

```
1171    // Delta = X + N_0_1* 10-blattW /10 *10 = X + N_0_1*(10-blatt)
1172    X += (int)(N_0_1*sigma);
1173
1174    X *= weights[player->getCurrentRound()];
1175    player->setSelbstsicherheitNervositaet(player->
            getEmotion_selbstsicherheitPanik()+X);
1176
1177  }
1178
1179  /**
1180   * Veraendert die Emotionen des Spieler m_player bei eigener Aktion: fold
1181   * Alle Informationen die benoetigt werden, wird vom EmoPlayer geholt.
1182   * Ueberschreibt die Funktion aus EmoStrategy.
1183   * Die Emotionsanpassung erfolgt anhand deterministischer Regeln, die von
           den Charakterwerten des Spieler und den Karten abhaengen.
1184   */
1185  void DeterministischeEmoAnpassung::ownFold()
1186  {
1187    EmoPlayer* player = getPlayer();
1188
1189    double sigma = (100- player->getCharacter_kartenBewerten())/30;
1190    int X = Kartentrans();
1191    double N_0_1 = normZZ()*sigma;
1192
1193    X = X + (int)N_0_1;
1194    int delta_freude = 0;
1195    int delta_nervoes = -40; //
1196    int delta_wut = 0;
1197
1198
1199     // (Pot/Anfangscash) ~ (bisher gesetzt/Anfangscash)
1200     // (bisher gesetzt)/Anf = (Anf - cash_now)/Anf = 1 - cash_now/Anf
1201
1202    double percent = (double)(1- (player->getMyCash() / player->
            getMyRoundStartCash()));
1203
1204    if(X<40) //schlechtes Blatt
1205     {
1206      int maxfreude = 50;
1207      int minfreude = -50;
1208      int low = 15; //Prozentwert: (gesetztes Geld / Geld am Anfang der
             Partie)
1209      int high = 80; //Prozentwert s.o.
1210
1211      if(percent>=low && percent<=high)
1212       {
1213        double x = ((maxfreude-minfreude)/(low-high)) * (percent - low) +
               maxfreude;
1214        delta_freude = (int)x;
1215       }
1216      else
1217       {
1218        if(percent<low) //wenig gesetzt -> gluecklicher
1219         {
1220          delta_freude = maxfreude;
1221         }
1222        else //viel gesetzt -> traurig
1223         {
1224          delta_freude = minfreude;
1225         }
1226       }
1227     }
1228    else
1229     {
1230      if(X>60) //gutes Blatt: wuetend -in relation- Pot/Anfangsgeld   ...
             traurig -in relation- Pot/Anfangsgeld
1231       {
1232        int maxwut = 50;
1233        int minwut = 0;
```

```
1234        int lowwut = 10;
1235        int highwut = 70;
1236
1237        int maxtrauer = -50;  //name: trauer da man bei einem guten Blatt
                eher traurig als gluecklich ist. unterschied soll in variable
                wiedergespiegelt sein
1238        int mintrauer = 0;
1239        int lowtrauer = 10;
1240        int hightrauer = 70;
1241
1242      if(percent>=lowwut && percent<=highwut)
1243       {
1244        double x = ((maxwut - minwut)/(lowwut-highwut)) * (percent - lowwut)
                + maxwut;
1245        delta_wut = (int)x;
1246       }
1247      else
1248       {
1249        if(percent<lowwut) // wenig gesetzt -> wenig wuetend
1250         {
1251          delta_wut = minwut;
1252         }
1253        else
1254         {
1255          delta_wut = maxwut;
1256         }
1257       }
1258      //das selbe fuer trauer
1259
1260      if(percent>=lowtrauer && percent<=hightrauer)
1261       {
1262        double x = ((maxtrauer-mintrauer)/(lowtrauer-hightrauer)) * (percent
                - lowtrauer) + maxtrauer;
1263        delta_freude = (int)x;
1264       }
1265      else
1266       {
1267        if(percent<lowtrauer) //wenig gesetzt -> gluecklicher
1268         {
1269          delta_freude = mintrauer;
1270         }
1271        else //viel gesetzt -> traurig
1272         {
1273          delta_freude = maxtrauer;
1274         }
1275       }
1276
1277
1278       } // end gutes Blatt
1279
1280      else // TODO: normales Blatt
1281       {
1282        int maxfreude = 30;
1283        int minfreude = -30;
1284        int lowfreude = 30; //Prozentwert: (gesetztes Geld / Geld am Anfang
                der Partie)
1285        int highfreude = 70; //Prozentwert s.o.
1286
1287       if(percent>=lowfreude && percent<=highfreude)
1288        {
1289         double x = ((maxfreude-minfreude)/(lowfreude-highfreude)) * (
                percent - lowfreude) + maxfreude;
1290         delta_freude = (int)x;
1291        }
1292       else
1293        {
1294         if(percent<lowfreude) //wenig gesetzt -> gluecklicher
1295          {
1296           delta_freude = maxfreude;
```

```
1297            }
1298           else //viel gesetzt -> traurig
1299            {
1300              delta_freude = minfreude;
1301            }
1302          }
1303
1304
1305
1306        } // end else: von normalem Blatt
1307      }
1308    player -> erhoeheNeutralHassGlobal(delta_wut);
1309    player->setTrauerEuphorie(player->getEmotion_trauerEuphorie()+
            delta_freude);
1310    //player->setSelbstsicherheitNervositaet(player->
            getEmotion_selbstsicherheitPanik()+delta_nervoes);
1311  }
```

# Abbildungsverzeichnis

# Literaturverzeichnis


[Aar97] E. J. Aarseth. *Cybertext : Perspectives on Ergodic Literature*. Johns Hopkins University Press, Baltimore, 1997.

[AFM02] U. Athenstaedt, H. H. Freudenthaler und G. Mikula. *Die Theorie sozialer Interdependenz*. Huber, Bern, 2002.

[AKA91] D. W. Aha, D. F. Kibler und M. K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991. `http://dblp.uni-trier.de`.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen und A. I. Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, page 307. AAAI/MIT Press, 1996. `DBLP,http://dblp.uni-trier.de`.

[Are04] M. Aretz. *Emotionen in der Künstlichen Intelligenz*. Diplomarbeit, FH Gießen-Friedberg, März 2004.

[Bae96] T. Baeck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, NY, USA, 1996.

[Bak87] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, New Jersey, USA, 1987. Lawrence Erlbaum Associates, Inc.

[Bar03] R. A. Bartle. *Design of Virtual Worlds*. New Riders, Indianapolis, 2003.

[BBD+03] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg und D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker, 2003.

[BDSS02] D. Billings, A. Davidson, J. Schaeffer und D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1-2):201–240, 2002.

[Bet80] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, University of Michigan, Ann Arbor, 1980.

[BH04] A. Bartsch und S. Hübner. *Emotionale Kommunikation - ein integratives Modell*. PhD thesis, Martin-Luther-Universität Halle-Wittenberg, 2004.

[BKKN95] C. Borgelt, F. Klawonn, R. Kruse und D. Nauck. *Neuro Fuzzy Systeme - Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen*. Vieweg, Wiesbaden, 1995.

[Bot95] H.-H. Bothe. *Fuzzy Logic - Einführung in die Theorie mit Anwendungen*. Springer, Berlin, Heidelberg, New York, 1995.

[Bra91] R. Brause. *Neuronale Netze - Eine Einführung in die Neuroinformatik*. B.G. Teubner, 1991.

[Bre08] C. Breazeal. Kismet. `http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html`, 2008. (Verfügbar 19.03.08).

[BS02] H. G. Beyer und H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[Bul08] F. Bulka. Dialog-KI in IT-basiertenSpielen Kommunikation zwischen Mensch und NPC. `http://bis.informatik.uni-leipzig.de/de/Lehre/0607/WS/GAMES/index`, 2008. (Verfügbar 19.03.08).

[Bur66] E. Burger. *Einführung in die Theorie der Spiele.* Walter de Gruyter & Co, Berlin, 1966.

[Cle98] J. Cleve. Wissensextraktion / data mining, 1998. `http://www.wi.hs-wismar.de/cleve/vorl/dm2160207807/skript/node1.html`.

[Csi05] M. Csikszentmihalyi. *Das flow - Erlebnis. Jenseits von Angst und Langeweile: im Tun aufgehen.* Klett-Cotta, Stuttgart, 2005.

[Dav91] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, NY, USA, 1991.

[DMW04] R. De Maria und J. L. Wilson. *High Score!: The Illustrated History of Electronic Games.* McGraw-Hill Osborne, New York, 2004.

[Dre61] M. Dresher. *Strategische Spiele - Theorie und Praxis.* Verlag Industrieller Organisation, Zürich, 1961.

[EF78] P. Ekman und W. V. Friesen. Facial action coding system: A technique for the measurement of facial movement. *Consulting Psychologists Press*, 1978.

[Ekm05] *Gefühle lesen*, Heidelberg, 2005. Elsevier, Spektrum Akad. Verl.

[FF03] J. Fritz und W. Fehr. *Zwischen Frust und Flow.* Bundeszentrale für politische Bildung, Bonn, 2003.

[FJS06] J. Feindt, C. Janßen und S. Sölbrandt. Gesprächskompetenz im I-can-EIB-System. In Claus Möbus, editor, *Web-Kommunikation mit OpenSource*, chapter 13, pages 147–212. Springer, Heidelberg, 2006.

[FK93] J. C. Fodor und T. Keresztfalvi. *Non-conventional conjunctions and implications in fuzzy logic.* Springer-Verlag, 1993.

[Flu06] M. Fludernik. *Einführung in die Erzähltheorie.* WBG Wissenschaftliche Buchgesellschaft, Darmstadt, 2006.

[Fog95] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, NY, USA, 1995.

[Fun07] J. Funke. Allgemeine Psychologie II: Emotion. Folien, 2007.

[GHJV04] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Entwurfsmuster.* Addison-Wesley, 2004.

[GR00] G. Goerz und C.-R. Rollinger. *Handbuch der künstlichen Intelligenz.* Oldenburg, 2000.

[Gra95] A. Grauel. *Fuzzy-Logik - Einführung in die Grundlagen mit Anwendungen.* Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich, 1995.

[Gra04] W. Grass. *Technische Anwendungen von Fuzzy-Systemen.* Universität Passau - Lehrstuhl für Rechnerstrukturen, Passau, 2004.

[GRS03] *Handbuch der Künstlichen Intelligenz.* Oldenbourg, 2003.

[Han07] N. Hansen. The CMA Evolution Strategy: A Tutorial. 2007.

[Heb49] D. Hebb. *Organization of Behavior.* Wiley, New York, 1949.

[Hof93] N. Hoffmann. *Kleines Handbuch Neuronale Netze.* vieweg, 1993.

[Hol71] R. B. Hollistien. *Artificial Genetic Adaption in Computer Control Systems.* PhD thesis, University of Michigan, Ann Arbor, 1971.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975. (Verfügbar 06.04.08).

[Hop82] J. J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *PNAS USA*, 79:2554–2558, 1982.

[Hor79] R. Horst. *Nichtlineare Optimierung.* Carl Hanser, München, 1979.

[Hui54] J. Huizinga. *Geschichte und Kultur.* Alfred Kröner Verlag, Stuttgart, 1954.

[Hui94] J. Huizinga. *Homo ludens. Vom Ursprung der Kultur im Spiel (1939).* Rowohlt Verlag, Hamburg, 1994.

[Jan05] T. Jansen. Evolutionäre Algorithmen. Vorlsungsskript WS 04/05, 2005.

[JJ98] D. W. Johnson und R. T. Johnson. Cooperative learning and social interdependence theory, 1998.

[Jue53] F. G. Juenger. *Die Spiele.* Klostermann Verlag, Frankfurt am Main, 1953.

[KDP83] S. Kirkpatrick, Gelatt C. D. und Vecchi M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Ken01] S. L. Kent. *The Ultimate History of Video Games: The Story behind the Craze that Touched Our Lives and Changed the World.* Three Rivers Press, New York, 2001.

[KMH+04] S. Kern, S. D. Mueller, N. Hansen, D. Bueche, J. Ocenasek und P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[Koh43] T. Kohonen. A logical calculus of the ideas immanent in nervous activity. *Ball Math Biophys*, 5:115–133, 1943.

[KW08] A. Koch und K. Woog. Emotionstheorie von Ortony, Clore & Collins und affektives Priming. `http://www.allpsych.uni-giessen.de/vf/WS-2006-seminar-emotion/PDF/11-woog-koch-ortony-priming.pdf`, 2008. (Verfügbar 19.03.08).

[KWDD98] M. Kirsten, S. Wrobel, F.-W. Dahmen und H.-C. Damen. Einsatz von Data-Mining-Techniken zur Analyse Ökologischer Standort- und Pflanzendaten. *Künstliche Intelligenz*, 12(2):39–42, 1998.

[Lam01] J. Lampel. Joebot - Einsatz von Neuronalen Netzen in einem Bot für Counterstrike. `http://johannes.lampel.net/bll137pub.pdf`, 2001.

[Lec06] M. Lechner. Automatische Textgenerierung in Computerspielen. `http://bis.informatik.uni-leipzig.de/de/Lehre/0607/WS/GAMES/index/\files?get=gsc_Marcus_Lechner.pdf`, 2006. (Verfügbar 20.02.08).

[LeD96] *The Emotional Brain: The Mysterious Underpinnings of Emotional Life*, 274, New York, 1996. Simon & Schuster.

[Lex07] Meyers Lexikonverlag. Verhalten, 02 2007.

[Lis02] K. Lischka. *Spielplatz Computer. Kultur, Geschichte und Ästhetik des Computerspiels.* Heise Verlag, Hannover, 2002.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th berkeley Symp. on Mathematical Statistics and Probability. 1967.

[Mai99] *Computernetze und virtuelle Realität: Leben in der Wissensgesellschaft.* Springer, 1999.

[Man03] T. Manninen. Interaction Forms and Communicative Actions in Multiplayer Games. *the international journal of computer game research*, 3(1), 2003. `http://www.gamestudies.org/0301/manninen/`, (Verfügbar 20.02.08).

[MBC+06] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley und C. H. Yong. *Computational Intelligence in Games Computational Intelligence: Principles and Practice.* IEEE Computational Intelligence Society, 2006.

[MdWS91] B. Manderick, M de Weger und P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150, San Mateo, 1991. Morgan Kaufmann.

[Mic86] R. Michalski. Understanding teh Nature of Learning. In T. M. Mitchell, editor, *Machine Learning – An Artificial Approach.* Morgan Kaufmann, Los Alto, CA, 1986.

[Mic92] Z. Michalewicz. *Genetic Algorithms + Data Structures =Evolution Programs.* Springer, Berlin, 1992.

[MK05] T. Manninen und T. Kujanpää. The Hunt for Collaborative War Gaming - CASE: Battlefield 1942. *the international journal of computer game research*, 5(1), 2005. `http://www.gamestudies.org/0501/manninen_kujanpaa/`, (Verfügbar 20.02.08).

[MM92] R. Maennde und B. Manderick. Parallel Problem Solving from Nature. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, North Holland, 1992.

[Mue92] H. Muehlenbein. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *PPSN*, pages 15–26, 1992.

[Mue07] I. Mueller. Emotionen beim Konsum von Bildschirmspielen. *Merz. Medien + Erziehung*, 4, 2007.

[Mun01] *Computer.Gehirn; Was kann der Mensch? Was können die Computer?*, Paderborn, 2001. Schöningh.

[Mur98] S. K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Min. Knowl. Discov.*, 2(4):345–389, 1998. `http://dblp.uni-trier.de`.

[Nis97] V. Nissen. *Einführung in Evolutionäre Algorithmen.* Vieweg, Wiesbaden, 1997.

[Oli05] F. Oliehoek. Game theory and ai: a unified approach to poker games, 2005.

[OS57] C. E. Osgood und P. H. Suci, G. J. und Tannenbaum. The measurement of meaning. *Chicago: University of Illinois Press*, 1957.

[OT90] A. Ortony und T. J. Turner. What's basic about basic emotions? In *Psychological Review*, 97, pages 315–331, 1990.

[PMAPA06] M. Ponsen, H. Munoz-Avila, Spronk P. und D. W. Aha. Automatically Generating Game Tactics with Evolutionary Learning. *AI Magazine*, 27(3):75–84, 2006. `http://www.cs.unimaas.nl/p.spronck/Pubs/ponsen_aimag.pdf`.

[PMASA05] M. J. V. Ponsen, H. Muñoz-Avila, P. Spronck und D. W. Aha. Automatically Acquiring Adaptive Real-Time Strategy Game Opponents Using Evolutionary Learning. In *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 1535–1540, Menlo Park, California, 2005. AAAI Press. `http://www.cs.unimaas.nl/p.spronck/Pubs/IAAI052PonsenM.pdf`.

[Pon04] M. Ponsen. *Improving Adaptive Game AI with Evolutionary Learning.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 2004. `http://www.kbs.twi.tudelft.nl/docs/MSc/2004/Ponsen_Marc/thesis.pdf`.

[PS04] M. Ponsen und P. Spronck. Improving Adaptive Game AI with Evolutionary Learning. In Q. Mehdi, N. Gough, S. Natkin und D. Al-Dabass, editors, *Computer Games: Artificial Intelligence, Design and Education*, pages 389–396, Wolverhampton, 2004. University of Wolverhampton. `http://www.cs.unimaas.nl/p.spronck/Pubs/PonsenCGAIDE.pdf`.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fommann-Holzboog, Stuttgart, 1973.

[Rei08] D. Reichardt. Emotional IT Computer mit Gefühl. `http://www.ba-stuttgart.de/fileadmin/ba/Dokumente/Akademietag_2004/Emotional_IT-Computer_mit_Gef_hl.pdf`, 2008. (Verfügbar 19.03.08).

[RHW86] D. E. Rumelhart, G. E. Hinton und R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323:533–536, 1986.

[RMS91] H. Ritter, T. Martinetz und K. Schulten. *Neuronale Netze - Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke.* Addison-Wesley, 1991.

[RN004] *Künstliche Intelligenz: Ein moderner Ansatz.* Pearson Studium, 2004.

[RSZ79] B. Rauhut, N. Schmitz und E.-W. Zachow. *Spieletheorie - Eine Einführung in die mathematische Theorie strategischer Spiele.* B.G. Teubner, Stuttgart, 1979.

[Rue08] G. Ruebenstrunk. Künstliche Gefühle. `http://www.ruebenstrunk.de/emeocomp/2.htm`, 2008. (Verfügbar 19.03.08).

[Sch68] H.-P. Schwefel. Experimentelle Optimierung einer Zweiphasendüse Teil I. Bericht Nr. 35 zum Projekt MHD–Staustrahlrohr 11.034/68, AEG Forschungsinstitut, Berlin, Oktober 1968.

[Sch77] H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research.* Birkhäuser, Basel, 1977.

[Sim67] H. A. Simon. Motivational and emotional controls of cognition. *Psychological Review*, 74:29–39, 1967.

[Sim83] H. A. Simon. Why Should Machines Learn? In T. M. Mitchell, editor, *Machine Learning – An Artificial Approach*, volume 1, chapter 3, pages 25–39. Morgan Kaufmann, Palo Alto, CA, 1983.

[SR95] H.-P. Schwefel und G. Rudolph. Contemporary evolution strategies. In *European Conference on Artificial Life*, pages 893–907, 1995.

[SS97] B. Sutton-Smith. *The Ambiguity of Play*. Harvard University Press, London, 1997.

[Sul01] J. Suler. The Psychology of Avatars and Graphical Space in Multimedia Chat Communities. In Michael Beiswenger, editor, *Chat Communication*, pages 305–344. Ibidem, Stuttgart, 2001. `http://www-usr.rider.edu/~suler/psycyber/psyav.html`, (Verfügbar 20.02.08).

[Sul07] J. Suler. Second Life, Second Chance. `http://www-usr.rider.edu/%7Esuler/psycyber/secondlife.html`, Januar 2007. (Verfügbar 20.02.08).

[Svo06] W. Svojanovsky. *Darstellung von Emotionen als Repräsentation von Dialogerfolg*. Diplomarbeit, Universität Karlsruhe (TH), November 2006.

[Tho07] J.-N. Thon. Kommunikation im Computerspiel. In Simone Kimpeler, Michael Mangold und Wolfgang Schweiger, editors, *Die digitale Herausforderung*, pages 171–180. VS, Wiesbaden, 2007.

[Tur50] A. M. Turing. Computing Machinery and Intelligence. *Mind*, 59:433–460, October 1950. One of the most influential papers in the history of the cognitive sciences: `http://cogsci.umn.edu/millennium/final.html`.

[Voe02] H. Voelz. Gefühle und Emotionen, 2002.

[Vog63] R. Vogelsang. *Die mathematische Theorie der Spiele*. Fred Dümmlers Verlag, Bonn, 1963.

[Wal06] C. Walter. Künstliche Emotionen. `http://www.dfki.de/~kipp/seminar/folien/Christine_Emotionen.pdf`, 2006. (Verfügbar 21.06.06).

[WBB02] T. Wright, E. Boria und P. Breidenbach. Creative Player Actions in FPS Online Video Games. *the international journal of computer game research*, 2(2), 2002. `http://www.gamestudies.org/0202/wright/`, (Verfügbar 20.02.08).

[Wei66] J. Weizenbaum. ELIZA–A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 1966. `http://i5.nyu.edu/%7Emm64/x52.9265/january1966.html`, (Verfügbar 01.03.08).

[Wei94] *Die Macht der Computer und die Ohnmacht der Vernunft*, 274, Frankfurt am Main, 1994. Suhrkamp.

[Wei07] K. Weicker. *Evolutionäre Algorithmen*. Teubner, Wiesbaden, 2007.

[Wer74] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, 1974.

[Wri91] A. H. Wright. Genetic Algorithms for Real Parameter Optimization. In Gregory J. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 205–218, San Mateo, CA, 1991. Morgan Kaufmann. `citeseer.ist.psu.edu/wright91genetic.html`.

[Yee02] N. Yee. Adolescent identities: Assemling the self. `http://www.nickyee.com/mosaic/adolescence.html`, 2002. (Verfügbar 20.02.08).

[Yee03a] N. Yee. Communication / Relationship Skills. `http://www.nickyee.com/daedalus/archives/000339.php`, 2003. (Verfügbar 20.02.08).

[Yee03b] N. Yee. Identity Projection. `http://www.nickyee.com/daedalus/archives/000431.php?page=1`, 2003. (Verfügbar 20.02.08).

[Yee06] N. Yee. VoIP Usage. `http://www.nickyee.com/daedalus/archives/001519.php?page=1`, 2006. (Verfügbar 20.02.08).

[Zad65] L. Zadeh. *Fuzzy Sets - Information and Control.* Academic Press, New York, 1965.