# Agenten-Verhaltensarchitekturen

Laura Dietz

26. Juni 2000

Seminar: „Techniken intelligenter Agenten“, Leiter: Dr. Brause, SS 2000

# Inhaltsverzeichnis

**1 Einleitung** **1**

**2 Verhaltensorientierte KI** **1**
2.1 Was ist ein intelligenter Agent? . . . . . 1
2.2 Was bedeutet „Agenten-Verhaltensarchitektur“? . . . . . 1
2.3 Die Subsumption Architektur . . . . . 2
2.4 Beispiel: Ein mobiler Roboter . . . . . 4
2.5 Vor- und Nachteile dieses Ansatzes . . . . . 5
2.6 Erweiterungen der Subsumption Architektur . . . . . 6

**3 Beispiele in Software** **7**
3.1 Softwareassistenten . . . . . 7
3.2 Prozessscheduling . . . . . 7

**4 Action Selection – Planen ohne Plan** **8**
4.1 Was ist Action Selection? . . . . . 8
4.2 Problemstellung . . . . . 8
4.3 Lösungsansätze . . . . . 8
4.4 Die Idee von Pattie Maes . . . . . 8
4.5 Beispiel: Ein Agent mit Hut . . . . . 10

**5 Lernen ohne globales Wissen** **11**
5.1 Wieso lernen? . . . . . 11
5.2 Lösungsansätze . . . . . 12
5.3 Offene Probleme . . . . . 12

**6 Zusammenfassung** **12**

# 1 Einleitung

Neben der traditionellen Künstlichen Intelligenz (KI), die auf der syntaktischen Analyse von Symbolen basiert, entstand ende der 80er Jahre die Verhaltensorientierte KI. Anstatt ein System nach funktionalen Gesichtspunkten, wie Wahrnehmung, Plangenerierung, Suchmechanismen und Spracherkennung zu zerlegen, kapselt man Teile, die ein Verhaltensmuster bewirken zusammen mit ihren Sensoren und ausführenden Einheiten zu einer Verhaltenskomponente. Es existiert keine globale interne Wissensrepräsentation, wie die Symbole der traditionellen KI, sondern man verwendet die Welt als ihr eigenes bestes Modell[1].

# 2 Verhaltensorientierte KI

## 2.1 Was ist ein intelligenter Agent?

Rodney Brooks gibt in [2] folgende umgangssprachliche Definition: „Im Feld der Künstlichen Intelligenz wird versucht, Computer Dinge tun zu lassen, die bei Menschen als Hinweise auf Intelligenz gelten." Solche Computer nennt man intelligente Agenten.

Agenten können als Roboter oder in Software auftreten, je nachdem, ob der Agenten in der physikalischen Welt oder in einer Cyberspace-Welt leben soll.

Agenten sollten autonom sein, d.h. selbstständig Probleme lösen und Entscheidungen treffen. Sie sollen zielorientiert und effektiv handeln, sich an dynamische Umgebungen adaptieren, bzw. robust auf unerwartete Zustände reagieren. Oft ist es erwünscht, dass sie neue Zusammenhänge erkennen, lernen und einsetzen können.

## 2.2 Was bedeutet „Agenten-Verhaltensarchitektur"?

Eine „Agenten-Verhaltensarchitektur" ist eine Architektur, die autonome Agenten nach der Verhaltensorientierten KI hervorbringt.

Nach [4] umfasst eine solche Architektur die Prinzipien und Organisationsstrukturen, die adaptivem, robustem und effektivem Verhalten unterliegen, sowie Entwicklungswerkzeuge, Techniken und Algorithmen, um Agenten nach diesen Prinzipien zu konstruieren.

Ein Hauptprinzip der verhaltensorientierten KI lautet: „Betrachtet man Problem, System und Umgebung zusammen, vereinfacht sich das Problem von selbst. Ebenso können aus einfachen Kommunikationsmechanismen komplexe Verhaltensweisen entstehen."[4]

Beispiele dafür sind:

- Ein lernendes System kann sich an bereits erprobte Pläne erinnern und muss daher keine hundertprozentig optimalen Pläne entwickeln.

- Ein Roboter mit visuellen Sensoren muss natürliche Sprache nicht genau verstehen, da es in dem Gespräch oftmals um Gegenstände aus der lokalen Umgebung geht. Gleiches gilt für Agenten, die Unsicherheiten durch Gegenfragen kompensieren.

- Ein Roboter, dem wenig Speicher zu Verfügung steht, kann anhand seiner Umgebung feststellen, welche Aufgaben er schon bewerkstelligt hat.

- Ein Roboter, der sich nur in normalen Büroräumen bewegt, kann sich darauf verlassen, Wände an horizontalen und vertikalen Kanten zu erkennen.

- Befindet sich der Agent in einer Multi-Agenten-Umgebung, die miteinander kommunizieren können, muss dieser Agent nicht für jedes Problem „das Rad neu erfinden", sondern kann bei anderen Agenten nachfragen, ob sie bereits eine Lösung gefunden haben.

Die Vertreter der „Verhaltensorientierten KI" kritisieren insbesondere den Ansatz der traditionellen KI, Module für einzelne Aufgaben wie Spracherkennung, Plangenerierung und Bildverarbeitung zu bauen, die nicht zusammenarbeiten. Dadurch wird viel Aufwand in hundertprozentige Lösungen gesteckt, wo ein ungenaueres Ergebnis, das meist schneller berechnet werden kann, ausreichen würde.

In der Verhaltensorientierten KI versucht man einfache Probleme in der echten Welt mit möglichst simplen Strukturen zu lösen. Es werden möglichst einfache Verhaltenskomponenten aufgebaut, die sich additiv zusammenfügen lassen. Häufig ist dann keine einzelne Verhaltenskomponente für das resultierende Verhalten zuständig. Beispiel dafür ist Matarics wall-following-robot [4], dessen Verhalten sich aus zwei Modulen ergibt: Das eine steuert auf eine Wand zu, wenn die Wand genügend weit entfernt ist, während das zweite den Roboter von der Wand wegsteuert, wenn sie zu dicht ist.

## 2.3 Die Subsumption Architektur

Die „Subsumption[1] Architektur" [3] wurde von Rodney Brooks entwickelt, um Roboter nach den Prinzipien der Verhaltensorientierten KI zu bauen. Dem zugrunde liegt das biologische Modell der Insekten.

Eine Verhaltenskomponente[2] besteht aus einem fest verdrahtetem Netzwerk aus erweiterten endlichen Automaten, den sogenannten AFSMs[3]. Jede AFSM besitzt verschiedene Zustände, ein oder zwei interne Register, einen internen Zeitgeber und Zugang zu einer einfachen Recheneinheit, die z.B.

[1] Übersetzt: Zusammengefassen
[2] Englisch: layer, behaviour oder competence module
[3] Augmented Finite State Machine

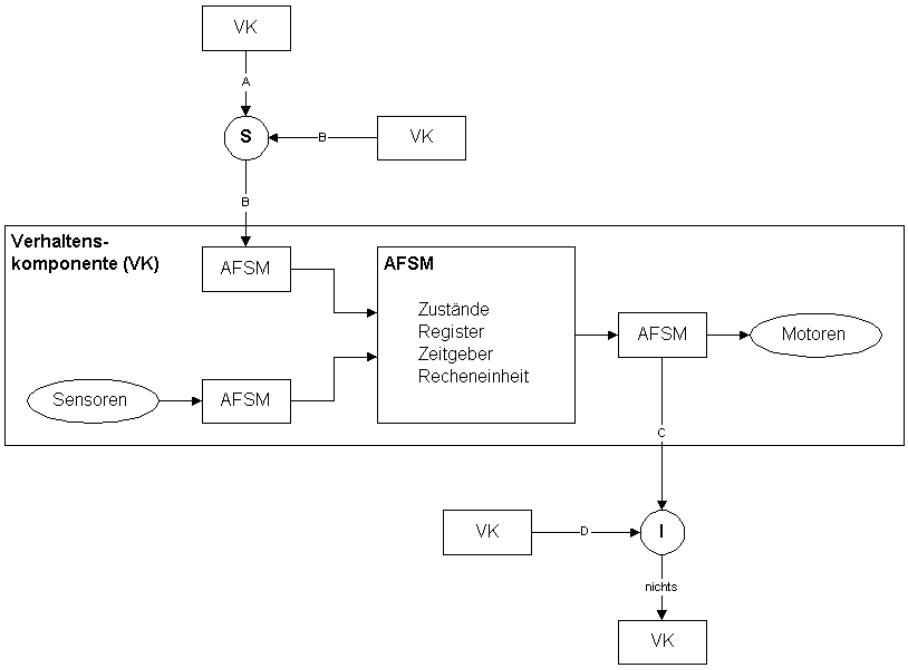


Abbildung 1: Aufbau der Subsumption Architektur

Vektorsummen berechnen kann. Die AFSMs arbeiten asynchron und können sich über Verbindungsdrähte kurze Nachrichten zuschicken.

Es gibt keine Kontrollzentrale. Die Arbeit der AFSMs beruht ausschliesslich auf dem dezentralten Nachrichtensystem. Durch die Ankunft einer Botschaft oder das Überschreiten einer Zeitspanne ändert eine AFSM ihren Zustand. Die AFSMs können die Nachrichten dekodieren oder weiterleiten, sie einem Prädikattest unterziehen, damit eine Berechnung anstellen und abhängig von ihnen ihren Zustand wechseln. Es gibt keine Möglichkeit auf globale Daten zuzugreifen oder Kommunikationskanäle dynamisch aufzubauen.

Für Verhaltenskomponenten gibt es einen Mechanismus, den Brooks „Suppression und Inhibition“[4] nennt. Möchte man eine neue Verhaltenskomponente zu einem System hinzufügen, klemmt man Datenleitungen einfach an bestehende Kabel an und ordnet ihnen Zeitkonstanten zu.

Bei „Suppression“ wird die neue Verbindung an die Eingangsseite der alten AFSM geklemmt. Trifft eine Nachricht über das neue Kabel ein, wird sie auf den Eingang der alten AFSM weitergeleitet, während Nachrichten, die über das ursprüngliche Kabel laufen, für den vordefinierten Zeitraum unterdrückt werden.

Bei „Inhibition“ wird das neue Kabel an den Ausgang der alten AFSM

[4]Übersetzt: Unterdrückung und Behinderung

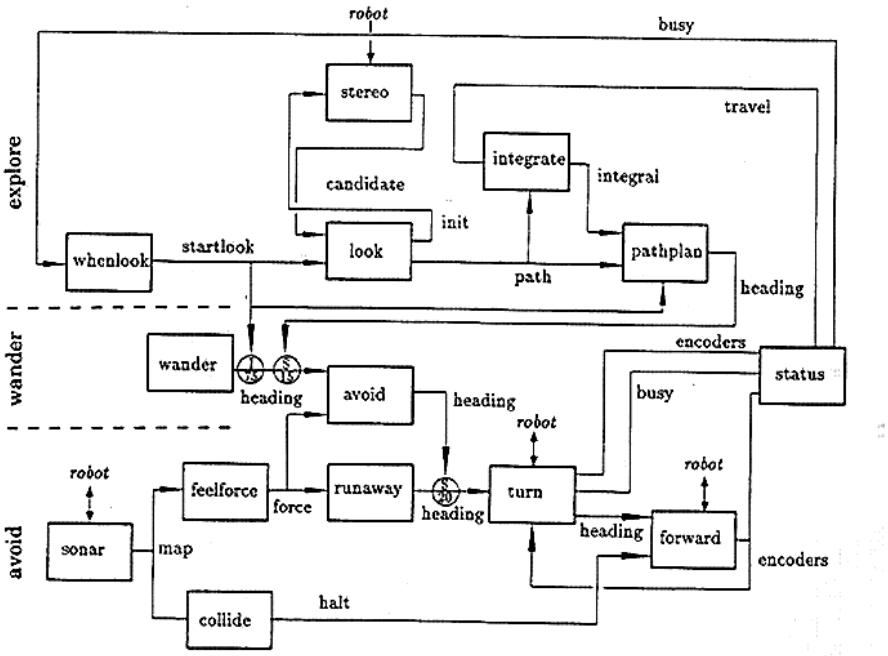


Abbildung 2: Aufbau der AFSMs von Roboter Allen mit seinen drei Verhaltenskomponenten [3]

gehängt. Durch eine Nachricht auf dem neuen Kabel, werden für den festgelegten Zeitraum ausgehende Meldungen der alten AFSM nicht weitergegeben. Im Gegensatz zur Suppression wird auch die Meldung des neuen Kabels nicht weitergeleitet, sondern dient nur als Verbots-Signal. Auf diese Weise kann eine AFSM ausgeschaltet werden.

## 2.4 Beispiel: Ein mobiler Roboter

Einer der ersten von Rodney Brooks gebauten Roboter ist Allen[1], dessen Umwelt die Büros des MIT sind. Er nimmt seine Umgebung durch mehrere Sonar-Sensoren wahr, welche eine Abstandsmessung durchführen. Sein Verhalten resultiert aus drei Verhaltenskomponenten: vermeide-Hindernisse, spaziere-durch-die-Gegend und laufe-zu-entfernten-Orten.

Vermeide-Hindernisse ist die älteste Komponente. Die Werte der Sonar-Sensoren liefern freie Weglängen, die dem Roboter sagen, in welche Richtung er ausweichen kann, wenn er ein statisches Hinderniss entdeckt oder sich etwas auf ihn zubewegt. Ein weiterer Reflex bringt den Roboter zum stehen, wenn sich etwas vor ihm in eine Richtung bewegt. Somit wartet er bis das Hindernis sich aus seiner Bahn herausbewegt hat.

In bestimmten Zeitintervallen „verspürt“ Allen den Wunsch sich in ei-

ne zufällige Richtung zu bewegen. Verantwortlich dafür die das „Spazier-Verhalten“. Zusammen mit der ersten Komponente wandert der Roboter somit durch die Gegend, vermeidet jedoch Kollisionen.

Die letzte Komponente bringt Allen dazu, entfernte Orte zu finden und dorthin zu gelangen. Dazu wird das Spaziere-durch-die-Gegend Verhalten unterdrückt. Durch die Kollisionsvermeidung driftet der Roboter jedoch von dem gewünschten Ziel ab. Diese Komponente muss also aufpassen, in welche Richtung sich Allen genau bewegt und korrigierende Kommandos senden.

Obwohl Sonarwerte naturgemäss extrem verrauscht sind, bewegte sich Allen durch die „natürliche“ Büroumgebung, ohne gegen Hindernisse zu laufen oder von ihnen getroffen zu werden.

## 2.5 Vor- und Nachteile dieses Ansatzes

Im Vergleich mit traditionellen Ansätzen liegen die Unterschiede in folgenden Punkten

- fehlende globale Wissensrepräsentation:
  - \+ kein Zeitverlust durch Übersetzung in Symbolwelt
  - \+ Daten unterschiedlicher Sensoren müssen nicht zu einem einheitlichen Modell vereinigt werden
  - \+ keine Suche in einer Datenstruktur
  - \+ stets Verwendung aktueller Werte
  - − nur Reiz-Reaktions-Verhalten
- dezentrales, paralleles System:
  - \+ kurze Antwortzeiten
  - \+ mehrerer Aufgaben gleichzeitig bearbeitbar
- keine ausgezeichnete Verhaltenskomponente ist für komplexes Verhalten verantwortlich:
  - \+ robustes, fehlertolerantes Verhalten
  - \+ leichtes Anpassen an verändernde Umgebung
  - − System nicht leicht zu verstehen
  - − keine Kontrollzentrale

Durch das Fehlen einer internen Wissensrepräsentation ist es schwierig, Verhalten zu erreichen, das nicht auf reinen Reiz-Antwort-Reaktionen basiert, wie etwa Planen oder zielorientiertes Handeln. Durch die feste Verdrahtung (insbesondere der Subsumption Architektur) ist es nicht offensichtlich, wie ein solches System lernfähig ist. Auf diese Fragen wird später näher eingegangen.

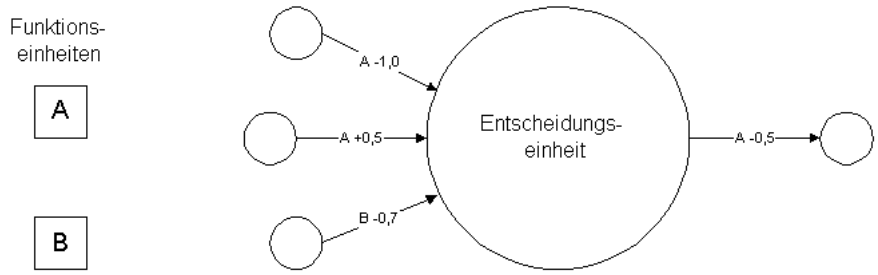


Abbildung 3: Bürokratisches Netzwerk aus Entscheidungseinheiten, vorgeschlagen von Rosenblatt und Payton

## 2.6 Erweiterungen der Subsumption Architektur

Man kann diese Netzwerke der AFSMs um globale Register und Monoflops, die AFSMs einer Komponente gemeinsam nutzen können, erweitern. Damit ist eine Synchronisation zwischen AFSMs leichter.

Pattie Maes [4] führt eine Aktivierungsenergie ein. Jede Verhaltenskomponente besitzt eine Menge an Energie, die sie an andere Verhaltenskomponenten weitergeben kann. Liegt die Gesamtenergie über einem Schwellenwert, so wird diese Komponente aktiviert. Dieser Ansatz ist eine allgemeine Variante des Suppression- und Inhibitionmechanismus, den Brooks vorschlägt.

Brooks erweiterte seine Architektur nachträglich um „Hormone“, mit denen es möglich ist, einen globalen Status zu verändern. Hormone sind Nachrichten, die durch das ganze Netzwerk fluten und Verhaltenskomponenten auf eine sehr allgemeine Weise aktivieren oder hemmen.

Rosenblatt und Payton[7] sehen als einen Nachteil der „Subsumption Architektur“, dass sich konkurierende Verhaltenskomponenten nur komplett unterdrücken können. Durch die strikte Blackbox-Trennung können einzelne Verhaltenskomponenten nicht auf den internen Status anderer Verhaltenskomponenten zugreifen, daher besteht keine Möglichkeit einen Kompromiss zu bilden.

Sie ersetzen AFSMs durch atomische Funktionselemente, die keinen internen Zustand haben, und verwalten sie, wie Brooks' Verhaltenskomponenten, durch ein Netz von Entscheidungseinheiten. Diesen werden von anderen Entscheidungseinheiten und externen Quellen gewichtete Möglichkeiten offeriert, unter denen sie eine Auswahl treffen, die sie weiterempfehlen oder ausführen.

# 3 Beispiele in Software

## 3.1 Softwareassistenten

Es soll ein Agent implementiert werden, der in Unix Systemen einem Benutzer bei seinen Eingaben auf der Shell hilft. Problematisch ist es, wenn der Benutzer häufig seine Verhaltensmuster ändert, mitten in einer Folge von Kommandos abbricht oder nicht-rationale Wege einschlägt.

Pattie Maes schlägt in [4] vor, die Verhaltenskomponenten so zu wählen, dass sie Experten für einzelne Kommandos sind. Diese beobachten die Eingaben des Benutzers, insbesondere in welchen Situationen „ihr“ Kommando aufgerufen wird. Erkennen sie so eine Situation wieder, schlagen sie ihr Kommando in gelernter Syntax vor.

Da eine Komponente nur einen sehr geringen Blickwinkel hat, kann sich dieses System leicht an veränderndes Benutzerverhalten anpassen. Auch ein kompletter Zusammenbruch ist unwahrscheinlich, da der Benutzer sein Verhalten nicht in allen Aspekten zugleich ändern wird.

## 3.2 Prozessscheduling

In einem Netzwerk mit mehreren Computern sollen Rechenjobs bearbeitet werden, so dass die Bearbeitungszeit eines Jobs möglichst gering ist. Jobs können zu beliebigen Zeitpunkten an verschiedenen Rechnern in Auftrag gegeben werden.

Ein Prozessmanager nach der traditionellen KI wäre in zentraler Weise strukturiert und würde die Informationen über die Last der einzelnen Computer sammeln und diese möglichst häufig aktualisieren. Das System könnte nicht weiterarbeiten, würde der zentrale Scheduler ausfallen.

Insbesondere Malone[5] schlägt ein marktplatzähnliches System vor, in dem Computer zu bearbeitende Jobs versteigern. Nicht ausgelastete Rechner schätzen für jeden angebotenen Job die Zeit, die sie dafür benötigen würden und geben dementsprechend ein Gebot für diesen Job ab. Der versteigernde Computer wartet eine kurze Zeit auf eingehende Gebote und vergibt dann den Job an einen Bietenden. Dieser Mechanismus wird nur für Jobs genutzt, deren Rechenaufwand höher ist als der Verwaltungsoverhead der Verteilung. Da es in diesem System keinen besonders kritischen Computer gibt, ist es wesentlich robuster als ein zentrales System: Ausgefallenen Rechnern werden keine Jobs zugewiesen. Allerdings muß für diesen Mechanismus die Bearbeitungszeit a priori bekannt sein.

[5] zitiert in [4]

# 4 Action Selection – Planen ohne Plan

## 4.1 Was ist Action Selection?

Action Selection befasst sich mit dem Auswählen einer Aktion, also der „Planung“ von Handlungsabläufen. Da es in der Verhaltensorientierten KI keine zentralen Kommando- oder Planungsmodule gibt, muss ein Mechanismus gefunden werden, wie sich die Verhaltenskomponenten das Kommando teilen können und dabei gemeinsam an einem Ziel arbeiten.

## 4.2 Problemstellung

Ein Agent befindet sich in einer Situation und soll bestimmte Ziele erreichen. Ihm stehen eine Reihe von Verhaltenskomponenten zu Verfügung. Wie können diese verteilten Verhaltenskomponenten koordiniert werden, damit die Ziele erreicht werden?

## 4.3 Lösungsansätze

Brooks Vorschlag ist, die Verhaltenskomponenten von Hand zu verdrahten. Durch Suppression und Inhibition könnten Verhaltenskomponenten, die den Zielen nicht zuträglich sind, ausgeschaltet werden. Leider plant das System dann nicht selbstständig.

Eine andere Idee ist es, eine Hierarchie einzuführen, in der Verhaltenskomponenten von einer Kommandoebene mitgeteilt wird, ob sie aktiv sein dürfen oder nicht.

Pattie Maes liefert in [5] eine alternative Methode, die bürokratische Koordinationsmodule vermeidet. Sie bewirkt zielorientiertes, situationorientiertes, an dynamische Umgebungen angepasstes, robustes und scheinbar vorrausschauendes Handeln.

## 4.4 Die Idee von Pattie Maes

Ein Agent besteht aus einer Menge an Verhaltenskomponenten und einprogrammierten Zielen. Jede Verhaltenskomponente wird durch ein Tupel aus Vorbedingungen, Bedingungen, die durch ihre Ausführung erfüllt würden und Bedingungen, die durch ihre Ausführung verhindert würden, repräsentiert. Damit eine Verhaltenskomponente ausführbar ist, müssen alle Vorbedingungen erfüllt sein.

Es wird ein Netzwerk generiert, durch das „Energie“ geleitet wird. Die „Energie“ sammelt sich in den Knoten, die der Ausführung der gegebenen Zielen am zuträglichsten sind.

Die Knoten des Netzwerks sind die Verhaltenskomponenten. Man verbindet nun Verhaltenskomponenten i und j durch

- Vorwärtskanten von i nach j, wenn die Ausführung von i eine Vorbedingung von j erfüllt,
- Rückwärtskanten von i nach j, wenn eine Vorwärtskante von j nach i existiert,
- Konfliktkanten von i nach j, wenn duch die Ausführung von i eine Vorbedingung von j unmöglich gemacht wird.

Man injiziert „Energie“ in das Netzwerk an allen Verhaltenskomponenten, die in der gegebenen Situation fast ausführbar sind oder deren Ausführung direkt zu einem gewünschten Ziel führen würde. Komponenten, die ein bereits erfülltes Ziel rückgängig machen würden, wird „Energie“ entzogen. Die „Energie“ verteilt sich folgendermaßen im Netzwerk:

- Jede Verhaltenskomponente, die ausführbar ist, gibt einen Teil ihrer „Energie“ an ihre Nachfolger weiter, da diese der Ausführung näher gekommen sind.
- Jede Verhaltenskomponente, die nicht ausführbar ist, gibt einen Teil ihrer „Energie“ an ihre Vorgänger weiter, da deren Ausführung jeweils eine ihre Vorbedingungen erfüllen würde. Damit der Mechanismus „vernünftige“ Ergebnisse erzielt, muss hier der Bruchteil größer sein als bei ausführbaren Komponenten.
- Jede Verhaltenskomponente entzieht Komponenten, die mit ihr in Konflikt stehen „„Energie“, in der Hoffnung, sie zu unterdrücken.

Ausgeführt, d.h. im Sinne von Brooks aktiviert, wird die Verhaltenskomponente,

- deren Vorbedingungen erfüllt sind
- deren „Energie“ einen Schwellenwert überschritten hat
- die unter diesen die meiste „Energie“ besitzt

Diese Methode scheint Verhalten hervorzubringen, das als „gut genug“ beschrieben wird. Durch Parameter wie Schwellenwert, Bruchteile der weitergegebenen „Energie“ (über Vorwärts- und Rückwärtskanten) und Menge der injizierten „Energie“ kann man das Verfahren nach seinen Ansprüchen steuern. Da sich dieses Verfahren nicht merkt, was es bereits wie getan hat und ob das gut war, kann es die gleichen Fehler immer wieder machen oder sich sogar in einem Kreislauf verfangen.

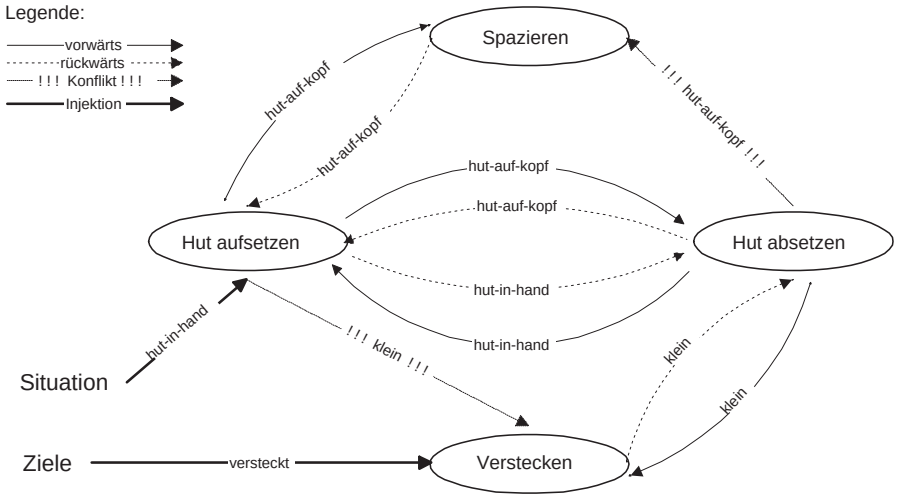


Abbildung 4: Generierter Graph zur Berechnung des Action Selection.

### 4.5 Beispiel: Ein Agent mit Hut

Aufgabenstellung: Es soll ein Agent gebaut werden, der einen Hut besitzt. Hat er den Hut auf dem Kopf, kann er spazieren. Will er sich verstecken, ist er mit Hut zu gross, muss ihn also absetzen.

Eine Architektur, die das Problem erfüllt besteht aus den vier Verhaltenskomponenten, Hut aufsetzen, Hut absetzen, Verstecken und Spazieren, mit folgenden Bedingungen:

| Vorbedingung | Verhaltenskomponente | | ermöglicht/verhindert |
|---|---|---|---|
| hut-in-hand | Hut aufsetzen | + | hut-auf-kopf |
| | | − | klein |
| hut-auf-kopf | Hut absetzen | + | klein |
| | | + | hut-in-hand |
| | | − | hut-auf-kopf |
| hut-auf-kopf | Spazieren | + | fortbewegen |
| klein | Verstecken | + | versteckt |

Gegebene Situation: Der Agent hält den Hut in der Hand.
Ziele: Sei versteckt.

Der Agent weiss nicht, dass er mit Hut in der Hand zum Verstecken klein genug ist. Dieses Beispiel zeigt, dass er trotzdem einen Weg findet, sich zu verstecken.

Algorithmischer Lösungsweg: Zuerst injiziert das System „Energie“ in „Hut aufsetzen“, da diese Vorbedingung erfüllt ist. Ein weiterer Injektionspunkt ist „Verstecken“, da dies unmittelbar zu dem Ziel „versteckt“ führt.

Da „Verstecken“ nicht ausführbar ist, läuft seine „Energie“ über Vorgängerkanten zurück zu „Hut absetzen“. Diese Komponente ist nur ausführbar, wenn „hut-auf-kopf“ erfüllt ist. Um dies zu erfüllen, läuft die „Energie“ weiter zurück zu „Hut aufsetzen“. Als alleinige Komponente ist „Hut aufsetzen“ ausführbar und hat „Energie“.

Nach ihrer Ausführung wird in „Hut absetzen“ und „Spazieren“ „Energie“ injiziert, da je eine ihrer Vorbedingungen erfüllt ist, und in „Verstecken“, da es das Ziel ist.

Die „Energie“ verteilt sich wie folgt:

- „Spazieren“ ist ausführbar und behält seine „Energie“.
- „Hut absetzen“ ist ausführbar und gibt einen Teil seiner „Energie“ an „Hut aufsetzen“ ab.
- „Verstecken“ ist nicht ausführbar und gibt einen Teil seiner „Energie“ an seinen Vorgänger „Hut absetzen“ weiter.

Daher ist die energetische Reihenfolge „Verstecken“, „Hut aufsetzen“, „Spazieren“, „Hut absetzen“. Ausführbar sind „Spazieren“ und „Hut absetzen“. Ausgeführt wird „Hut absetzen“, da es mehr „Energie“ als „Spazieren“ besitzt.

Durch erfüllte Vorbedingungen erhalten in der nächsten Iteration „Hut aufsetzen“ und „Verstecken“ „Energie“. Anschliessend bekommt „Verstecken“ eine weitere Portion, da es zum Ziel führt. „Verstecken“ entzieht „Hut aufsetzen“ über eine Konfliktkante „Energie“. Anschliessend wird „Verstecken“ ausgeführt.

Nachdem nun das Ziel „versteckt“ erfüllt ist, würden weiterhin alle Verhaltenskomponenten, die dieses Ziel verhindern würden, zusätzlich unterdrückt werden.

# 5 Lernen ohne globales Wissen

## 5.1 Wieso lernen?

Viele der bestehenden verhaltensorientierten Ansätze sind robust und flexibel. Nur wenige profitieren jedoch aus Fehlern und Erfolgen oder können gar Wissen über ihre Umwelt anhäufen.

Weiterhin wird es als sehr schwierig beschrieben, komplexe Agenten fehlerfrei zu implementieren. Die Idee ist, Agenten mit primitivem Verhalten und wenig Grundwissen in ihre Welt zu bringen. Dort soll der Agent durch Experimentieren Zusammenhänge erkennen und lernen. Zum Beispiel könnte das von Pattie Maes vorgeschlagene System zum Action Selection feststellen, dass der schon einmal eingeschlagene Weg schlecht war, und ihn zusätzlich unterdrücken.

### 5.2 Lösungsansätze

Da in verhaltensorientierten Architekturen weder eine zentrale Wissensrepräsentation noch ein explizites Lernmodul existiert, muss ein Lernmechanismus von allen Verhaltenskomponenten unterstützt werden. Lösungsansätze müssen daher dezentralisierbar sein und sich in das Modell der Subsumptionnetzwerke integrieren.

[4] verweist auf drei Ansätze: Reinforcement Learning[6], Klassifizierungssysteme und Modell-Lernen.

Der Grundansatz der ersten beiden ist, nach einer Aktion auf ein Signal zu warten, das dem Agenten verrät, ob die Aktion gut oder schlecht war. Es wird versucht die Zuordnung $(Situation, Aktion) \mapsto Belohnung$ zu bestimmen. Anschliessend wird die Aktion bevorzugt, die in der gegebenen Situation die höchste Belohnung eingebracht hat.

Agenten nach dem Modell-Lern-Prinzip führen zufällig Experimente aus und beobachten, welche neue Situation entsteht. Dadurch bestimmen sie die Zuordnung $(Situation, Aktion) \mapsto Situation$. Durch gelernte Zuordnungen wird die Wahl von zukünftigen Aktionen beeinflusst. Die Tabelle zu dem von Pattie Maes vorgeschlagenem Planungsmechanismus könnte auf diese Weise erweitert werden.

### 5.3 Offene Probleme

Gerade mit fehlerhaften oder verrauschten Sensoren ist es schwierig, die aktuelle Situation zu erkennen. Ein System, dass nichts über seine Umwelt weiss, hat es schwer, Identifikationsmerkmale für Situationen zu finden.

Entsteht eine neue Situation durch das gleichzeitige Ausführen von Aktionen, externen Einflüssen oder mit einer zeitlichen Verzögerung, ist es problematisch die verantwortlichen Aktionen ausfindig zu machen.

Ausserdem sind Reinforcement Learning und Klassifizierungssysteme auf feste Ziele beschränkt und es ist nicht möglich, dem Agent ein initiales Wissen mitzugeben.

## 6 Zusammenfassung

In Verhaltensorientierten Architekturen steht der Reiz-Reaktions-Gedanke im Vordergrund. Man baut auf ethologischem Wissen über Insekten und primitiven Tieren auf.

Im Gegensatz zur Dekomposition eines Systems nach funktionalen Gesichtspunkten interner Arbeitsweise, wie in traditionellen Systemen üblich, findet die Unterteilung in Verhaltenskomponenten nach externen Anforderungen statt. Eine Verhaltenskomponente entspricht dabei einer „Black

[6] Übersetzt: Lernen durch Verstärkungssignale

Box“, die über Datenkanäle mit anderen Verhaltenskomponenten kommunizieren kann.

Anstatt, wie in der traditionellen KI, eine globale Wissensrepräsentation zu verwenden, teilen Verhaltenskomponenten sich das Kommando über primitive Schichten. Sensoren und ausführende Hardware sind eng gekoppelt, so dass nur relevante Fakten verarbeitet werden. Dies hat den Effekt, dass Engpässe (wie zentrale Suche) vermieden werden und eine parallele, dezentrale Arbeitsweise ermöglicht werden kann. Verhaltensorientierte Agenten sind im Allgemeinen robust, da durch redundante Sensorinformationen leicht fehlerhafte Daten ausgeglichen werden.

Offene Fragen sind, ob sich zunehmend komplexe verhaltensorientierte Systeme bauen lassen, die performant sind, und wie man höhere Intelligenzlevel implementieren kann. Häufig kann man kein explizites Modul für das Gesamtverhalten verantwortlich machen. Es ist daher schwierig, diese Systeme zu verstehen. Von Anhängern der traditionellen KI wird bemängelt, dass oft nur an konkreten Problemen gearbeitet wird, anstatt eine globale Lösung zu suchen, so dass für leicht geänderte Problemstellungen die alte Lösung nur schlecht wiederverwendbar ist.

Hauptinitiatoren dieser Richtung sind Rodney Brooks und Pattie Maes. Rodney Brooks ist der Begründer der Subsumption Architektur und beschäftigt sich mit dem Bau von Robotern. Pattie Maes liefert ein Pendant für Softwareagenten und entwickelt Mechanismen für Action Selection und Lernen.

## Literatur


[1] Rodney A. Brooks: *Elephants Don't Play Chess*, in: Robotics and Autonomous Systems Vol. 6, 1990, pp 3 - 15, http://www.ai.mit.edu/people/brooks/papers/elephants.ps.Z

[2] Rodney A. Brooks: *New Approaches to Robotics*, in: Science Vol. 253, Sept 1991 pp 1227 - 1232, http://www.ai.mit.edu/people/brooks/papers/new-approaches.ps.Z

[3] Rodney A. Brooks: *Intelligence without representation*, 1987, in: Artificial Intelligence 47 (1991), p 139 - 159, http://www.ai.mit.edu/people/brooks/papers/representation.ps.Z

[4] Pattie Maes: *Modeling Adaptive Autonomous Agents*, in: Artificial Life Journal Vol. 1, An Overview, MIT Press, Cambridge, 1994, http://pattie.www.media.mit.edu/people/pattie/alife-journal.ps

[5] Pattie Maes: *How To Do The Right Thing*, in: Connection Science Journal, Vol. 1, 1989, http://agents.www.media.mit.edu/groups/agents/Publications/Pattie/consci/consci.ps

[6] Pattie Maes: Notizen zur Veranstaltung *MAS 738 – Modeling Autonomous Agents, Spring 1996*, in: http://pattie.www.media.mit.edu/people/pattie/class-agents/Slides-MAS738-Feb16.html

[7] J. Kenneth Rosenblatt und David W. Payton: *A Fine-Grained Alternative to the Subsumption Architecture*, in: In Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, Wahsington DC, June 1989, Vol. 2, pp 317 - 324, IEEE Press, 1989 http://www.umiacs.umd.edu/~julio/papers/Fine_Grained_Alternative.ps.gz

[8] Robert Wray, Ron Chong, Joseph Phillips, Seth Rogers und Bill Walsh: *A Survey of Cognitive and Agent Architectures*, 1992, (lose Informationssammlung, nur im Internet verfügbar) http://ai.eecs.umich.edu/cogarch0/index.html