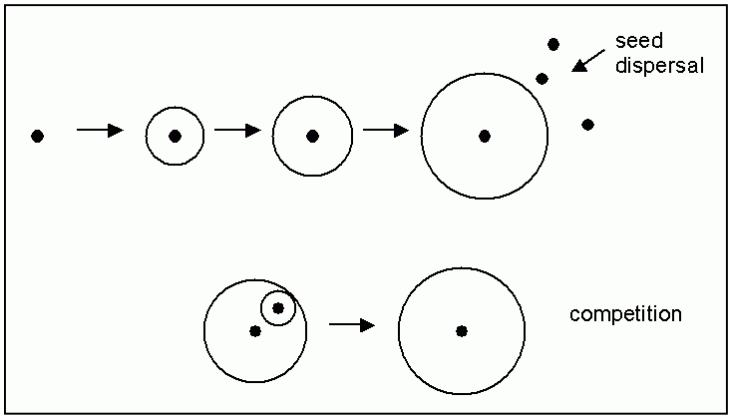
4. Aktivität von Organismen in ihrer Umgebung

Aktivität:
- Reaktivität
- Eigenaktivität

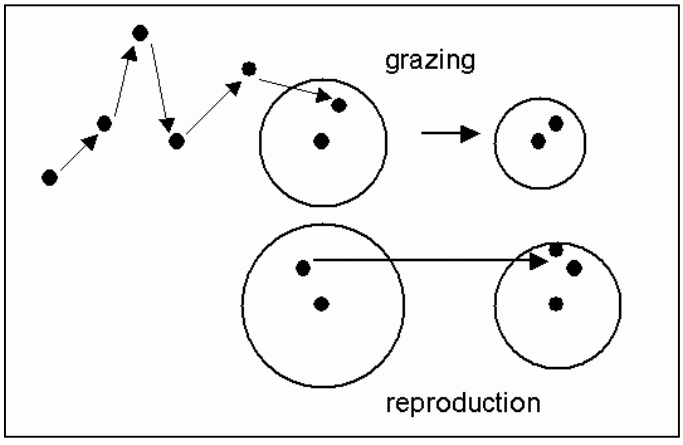
einfache Aktivitätsmuster lassen sich schon in die bisher behandelten formalen Modelle einbauen:
- Reaktivität durch Abfrage von Bedingungen (Verzweigungen; case-Anweisungen)
- Eigenaktivität durch Zufallskomponenten

Beispiel:
L-System-Modell eines einfachen "Ökosystems" aus Pflanzen und Tieren

Aktivitäten der Tiere:
- Nahrungssuche (Suche nach Pflanzen; *random walk*)
- Fressen (dabei Bewegung vermindert)



Verhalten d. Pflanzen



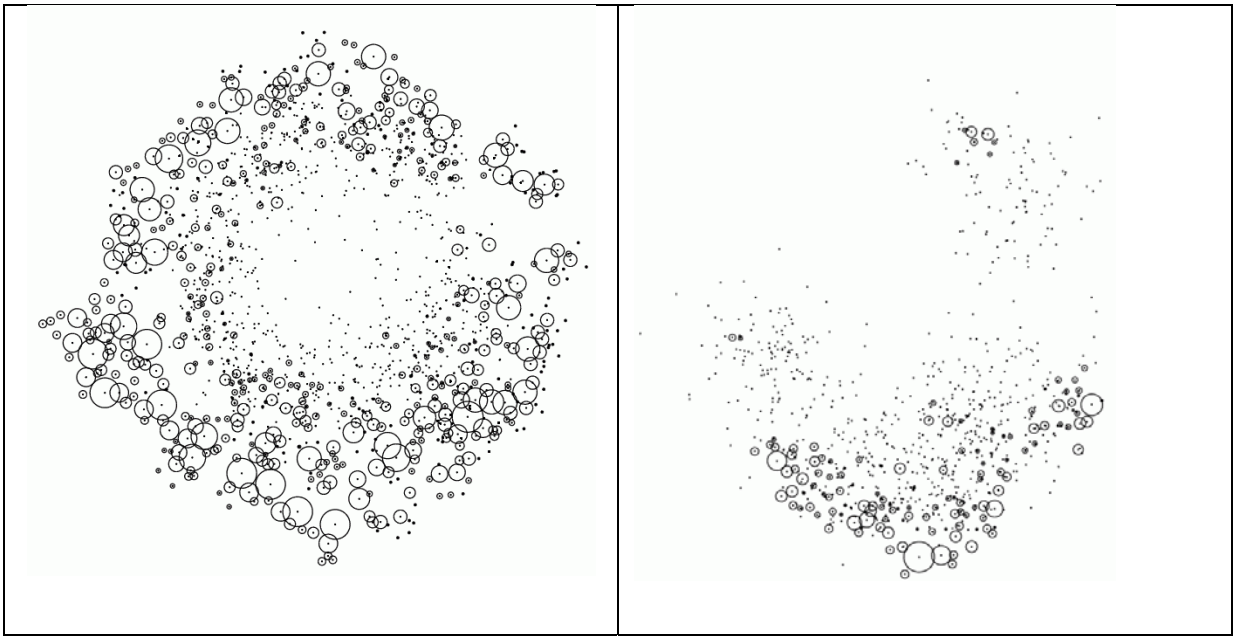
Verhalten der Tiere

L-System (Grogra-Syntax):

```
/* Parameters: */
\const lag 15,
\const pgrow 0.9,
\const init 4,
\const respi 0.25,
\const eat 1.0,
\const thr 7.5,
\const short 0.4,
\const init_w 15,
\var eps uniform -1 1,
\const pmaxage 30,
\const pgenage1 10,
\const pgenage2 18,
\const pminrad 9,
\const pgenfac 0.3,
\const ang 45,
\var ran uniform 5 15,
\var rr uniform 0 360,
\var dist uniform 15 40,
\var i index,
\var rad local 0,
\var len length,
\angle 90,
\var f function 21 1,
\var sh function 4 1,
\var control function 30 2,
\const acol 14,  /* 2 */
\const pcol 3,   /* 3 */
\const ocol 1,
\const adcol 5,
\var gn generation,
\var out function 30 1,
\askrandomseed,
\axiom circ 1,
\askaxiom,

/* Initialization: */
* # [ RH(rr) + f(ran) - a(-lag, init, init_w) ] P(pcol) p(0,0) M(12),
(t < 0)  a(t,e,w) # a(t+1, e, w),

/* Behaviour of animals: */
(e <= 0) a(t,e,w) # ,
(e > thr) a(t,e,w) # [ RH(rr) + f(short) - a(0, e/2 - respi,
                                             max(0, w+eps)) ]
                       RH(rr) + f(short) - a(0, e/2 - respi,
                                             max(0, w+eps)),
(f(pcol) > 0) a(t,e,w) # RH(rr) + f(short) -
                         a(t+1, e + eat - respi, w) Ar+(rad, -eat),
Ar+(x, y) # ,
a(t,e,w)  # RH(rr) + f(w) - a(t+1, e - respi, w),

/* Behaviour of plants: */
(t > pmaxage) p(t,r) # ,
(r < 0)       p(t,r) # ,
(sh(ang) > 0) p(t,r) # ,
(((t=pgenage1) || (t=pgenage2)) && (r >= pminrad))
   p(t,r) # &(pgenfac*rad) < [ RH(rr) + f(dist) - p(0,0) ] >
            p(t+1, rad),
p(t,r) # p(t+1, rad + pgrow),
```

```
/* interpretive rules: */
a(t, e, w) ## Pl(if(t<0, adcol, acol)) Dl(0.2*e) C(w) F(0),
p(t,r) ## L(r) [ Pl(ocol) Dl1 O(circ, 1) ]
          f(r) RU180 N(100) Dl(0.1) F KL(rad),
circ ## P(pcol) + &(30) < [ f1 S(i) ] RL12 >
          &(29) < C(i,i+1) > C(29,0)
```

2 Simulationsergebnisse mit verschiedenen Parametersätzen:



Typen einfacher Verhaltensweisen (gewonnen am Beispiel der Bewegung von Insekten – Beer et al. nach Thro 1994):

Reflex	sofortige Reaktion auf plötzlichen Reiz (beim Menschen unbewusst)
Taxis	Orientierung aufgrund von Gradienten in der Umgebung (Licht, Schall, chem. Konzentration, Schwerkraft...) – zur Quelle hin oder von ihr weg
Reak. auf angeborenen Stimulus	Auslösung von Verhaltensweisen durch festes Schema (Flucht vor Raubtier-Silhouette; Kindchenschema...)
Appetenzverhalten	ausgelöst durch Kombination von Reiz und einem inneren Zustand (z.B. Fressen – bei Vorhandensein von Futter und Hunger)
bedingter Reflex	Verhaltensänderung als Folge von Reizwiederholung

Einfache Typen von Bewegungsverhalten für die Robotersteuerung (nach Anderson & Donath; Thro 1994):

Anziehung nach vorn
bewegt den Roboter auf seinem aktuellen Kurs

Ortsanziehung
Roboter geht zu einem bestimmten Standort im Raum (vgl. Zugverhalten der Vögel)

Anziehung durch ein Objekt
Roboter bewegt sich auf ein von ihm ermitteltes Objekt zu. Wenn kein Objekt ermittelt, keine Bewegung.

Folge Objekt gegen den Uhrzeigersinn [im Uhrzeigersinn]
Roboter dreht sich um das Objekt

Anziehung durch begrenzte Räume
bewegt den Roboter in das Gebiet, das sich im kleinsten Winkel öffnet. Dies ermöglicht Schutz vor "Feinden", die zu groß sind, um in diesen Raum hineinzupassen

Anziehung durch offene Räume
bewegt den Roboter in das Gebiet, das sich im größten Winkel öffnet. Nutzen: bestmöglicher Überblick über die Umgebung

Ortsgebundene Anziehung durch offene Räume
Roboter bewegt sich durch offene Gebiete, die (ungefähr) in der gleichen Richtung liegen wie der Zielort

Passive Vermeidung
Erstarren auf der Stelle, wenn eine Kollision mit einem Objekt droht

Aktive Vermeidung
Roboter weicht dem Objekt aus

Durch Kombination einiger dieser Verhaltensweisen entstehen neue Verhaltensweisen:

Anziehung nach vorn + Aktive Vermeidung + Passive Vermeidung bewirkt *generelles Herumwandern*, bei dem der Roboter einen großen Teil seiner Umgebung durchstreift.

Aktive Vermeidung + Passive Vermeidung + Ortsanziehung bewirkt *einfache Navigation*, bei der sich der Roboter ohne Kollision mit irgendeinem Objekt auf sein Ziel zubewegt.

Aktive Vermeidung + Ortsanziehung bewirkt *Stagnation*, die ähnlich wie einfache Navigation ist, ihre einfachsten Verhaltensweisen neutralisieren sich jedoch gegenseitig. Der Roboter hält an, kurz bevor er die angepeilte Position erreicht hat.

Passive Vermeidung + Aktive Vermeidung + Anziehung durch ein Objekt + Folge Objekt (im und gegen den Uhrzeigersinn) bewirkt das *Folgen an der Peripherie*, bei dem der Roboter sich dicht an umgrenzenden Mauern und anderen Objekten der Umgebung entlangbewegt.

Anziehung durch offene Räume + Aktive Vermeidung + Passive Vermeidung + Anziehung nach vorn bewirkt das *Durchwandern weit offener Räume*, bei dem der Roboter durch das Zentrum dieser Räume marschiert.

Anziehung durch begrenzte Räume + Aktive Vermeidung + Passive Vermeidung + Anziehung nach vorn bewirkt das *Durchwandern eng begrenzter Räume*, bei dem sich der Roboter in der Nähe der Öffnung bewegt.

Ortsgebundene Anziehung durch offene Räume + Aktive Vermeidung + Passive Vermeidung + Anziehung nach vorn bewirkt die *ortsgerichtete Durchwanderung offener Räume*, die ein zyklisches Muster darstellt. Der Roboter bewegt sich auf seinen Zielort zu, wobei er in einem offenen Raum anhält und dann weitermarschiert. Dieses Muster wiederholt sich mehrere Male, bevor er den angestrebten Ort erreicht.

Wie können Verhaltensregeln verknüpft und repräsentiert werden?

Klassifizierungssystem (John Holland):

- Menge von Regeln (*Classifyern*)
  jede codiert durch 2 Bitstrings: z.B. 01# : 110 (# = "don't care")
  oder durch je 2 Symbolfolgen
- Detektoren, um Fakten aus der Umwelt wahrzunehmen
- diese werden als Symbolfolgen codiert und auf eine "Anzeigetafel" gesetzt
- wenn die Nachrichten auf der Tafel die Bedingungen der Klassifizierungsregeln erfüllen, erfolgt eine Aktion: neue Nachricht wird auf die Tafel gesetzt; ggf. werden zusätzlich "Effektoren" für Außenwirkung aktiviert
- die Regeln werden durch Stärke-Indices gewichtet

Beispiel "Frosch":

WENN kleines fliegendes Objekt von links, DANN sende @.
WENN kleines fliegendes Objekt von rechts, DANN sende %.
WENN kleines fliegendes Objekt in der Mitte, DANN sende ž .
WENN großes näherrückendes Objekt, DANN sende!.
WENN kein großes näherrückendes Objekt, DANN sende *.
WENN * und @, DANN bewege den Kopf 15° nach links.
WENN * und %, DANN bewege den Kopf 15° nach rechts.
WENN * und ž , DANN bewege dich in die Richtung, in die der Kopf zeigt.
WENN !, DANN bewege dich schnell in die entgegengesetzte Richtung, in die der Kopf zeigt.

Anzeigenbrett

(Enthält die jüngsten Nachrichten)

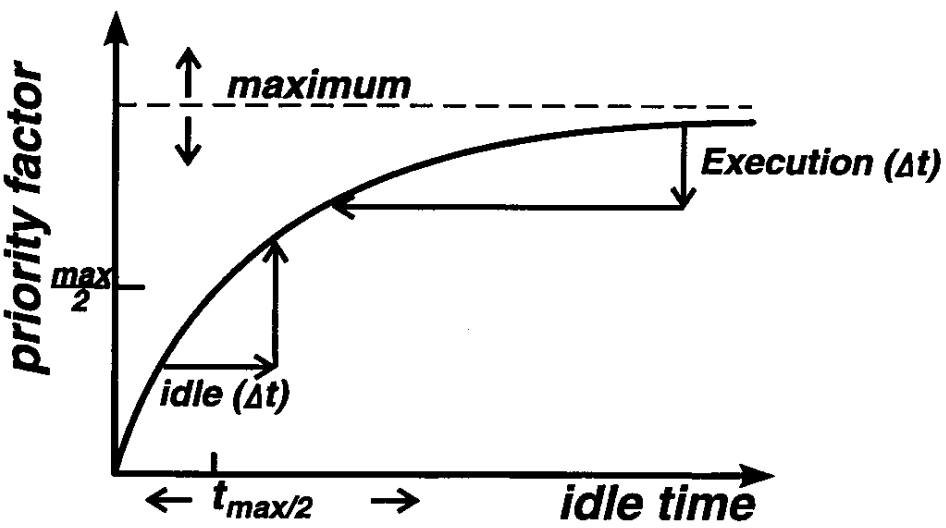
* @ .

Individuenbasierte Tiermodelle (Etho-Modelling)

Beispiel: Rotkehlchenmodell von Reuter & Breckling (Breckling et al. 1997)

Tiere als Objekte i. Sinne der OOP
Variablen für jedes Individuum: Position, Gewicht, Energievorrat, Territorialbesitz ...
Methoden (Tasks): Singen, Jagen, Ruhen, Junge füttern...

- jede Aktion kostet Energie
- Verhaltensregeln verknüpfen Bedingungen mit den Methoden
- Tasks sind mit veränderlichen Prioritätsfaktoren versehen



- Auswahl einer Task zu einem gegebenen Zeitpunkt resultiert aus: Umwelteinflüssen, innerer Zustand (z.B. Energiebedarf; aktuelle Phase im Brutzyklus), Interaktion mit anderen Rotkehlchen (Revierverhalten; Paarung; Brutpflege) und mit Raubtieren
- durch *time scheduling* (kontinuierliche Zeit; in SIMULA inhärent) wird für alle Individuen simultan der Aktivitätszustand verwaltet
- die Parameter beruhen auf gemessenen Daten!

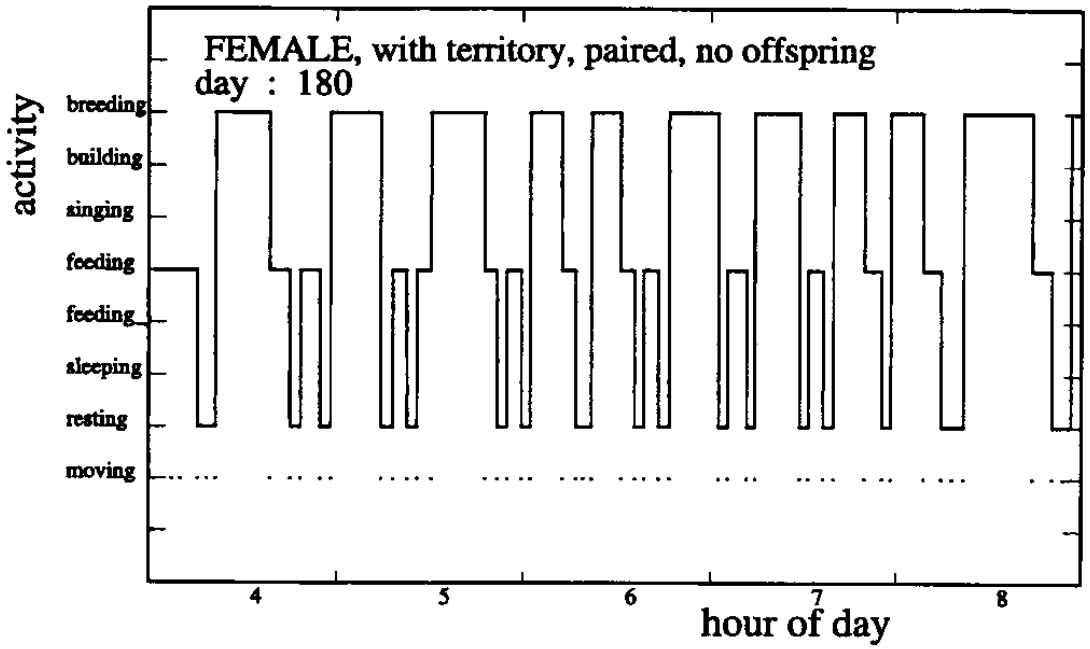


Figure 7a. Activity sequence for a breeding female. Results from the simulation program.

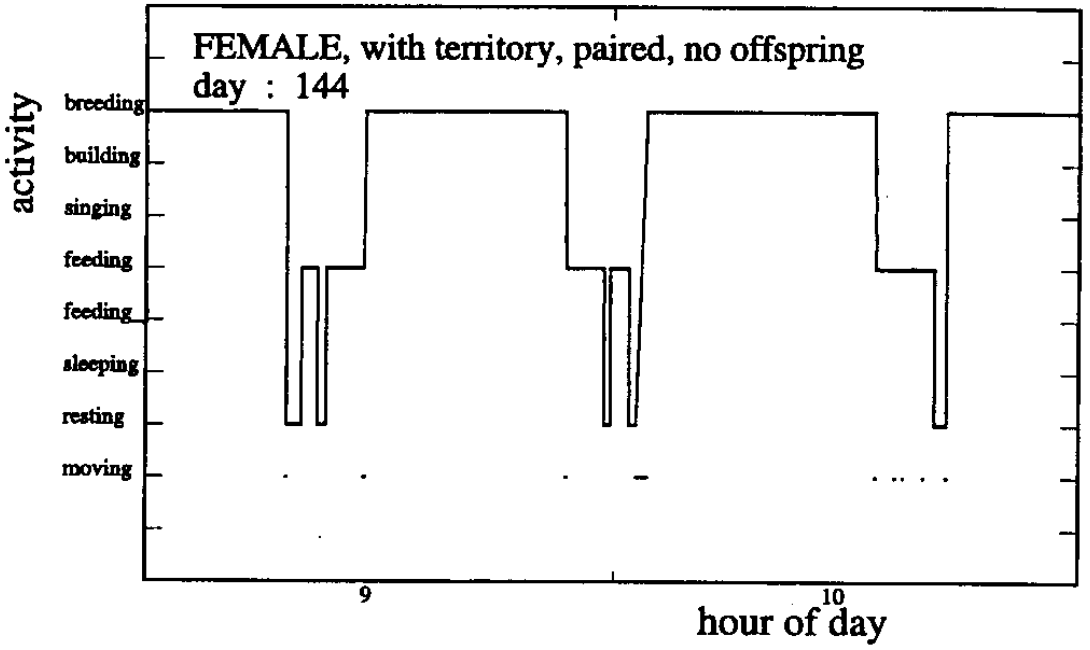


Figure 7b. Activity sequence for a breeding female. Field data (personal communication, B. Grajetzky, Project Center for Ecosystems Research, unpublished data).

Simulationsergebnis (oben) und gemessene Daten (unten)

(aus Breckling et al. 1997)

Von höherentwickelten Tieren erwarten wir, dass sie nicht nur nach einem festen Regelsystem aktiv sind, sondern frühere Erfahrungen einbeziehen in die Steuerung ihres Verhaltens – dass sie *lernen*

Was ist Lernen?

8 Lerntypen nach Gagné:

Typ 1: *Signallernen* (bedingter Reflex; Pawlowsche Konditionierung)
Futter → Speichelfluss
Futter + Glockenton → Speichelfluss (wiederholt)
Glockenton → Speichelfluss

Typ 2: *Reiz-Reaktions-Lernen* (Stimulus-Response; Skinner'sches Lernen)
Lernen durch Verstärkung (positiv: Belohnung, negativ: Bestrafung); Verhalten wird häufiger, wenn eine Verstärkung damit verknüpft wird
- funktioniert auch (sogar besser!) bei "intermittierender" (unzuverlässiger) Verstärkung

Typ 3: *Kettenbildung / motorische Ketten*
Verbindung einer Abfolge motorischer Reiz-Reaktions-Verhaltensweisen
Kompetenzlernen, z.B. Radfahren, Schwimmen, eine Suppe zubereiten...

Typ 4: *Kettenbildung / sprachliche Assoziation*
Verbindung einer Abfolge verbaler Reiz-Reaktions-Verhaltensweisen
Beispiele: Zählen, Gedicht aufsagen

Typ 5: *Lernen multipler Diskriminationen*
Lernen, zwischen hochgradig ähnlichen Reizinputs zu unterscheiden
z.B. gleich klingende Wörter in verschiedenen Sprachen

Typ 6: *Begriffslernen*
Ordnen von Dingen zu Klassen, Reagieren auf Klassen als Ganze
z.B. Begriffe "Hund", "Mensch"

Typ 7: *Regellernen*
"Regel" hier als eine erschlossene Fähigkeit, die das Individuum befähigt, auf eine Klasse von Reizsituationen mit einer Klasse von Leistungen zu reagieren

Typ 8: *Problemlösen*
Anwendung mehrerer Regeln bringt Regeln höherer Ordnung hervor
z.B. Strategien beim Schachspiel

nicht zu diesem Schema passend (z.T. quer liegend):
weitere Lerntypen

Prägung: Erlernen eines komplexen Musters in einer sensitiven Phase (K. Lorenz – Graugänse-Küken lernen, Mutter zu erkennen)

Imitationslernen (oft bei Kettenbildung (Typ 3, 4) beteiligt, aber evtl. auch bei höheren Typen)

protokollarisches Lernen: Aufnahme von Ereignissen ins Gedächtnis
man unterscheidet:

- sensorisches Gedächtnis (hohe Kapaz., ca. 1 Sek.)
- Kurzzeitgedächtnis (nur ca. 7 Items, 10 Sek.)
- Langzeitgedächtnis (unbegrenzt)

Priming-These: Alle Gedächtnisinhalte sind verbunden mit den Umweltbedingungen während der Informationsaufnahme (z.B. Gerüche, Körperhaltung, Räumlichkeiten etc.)
Vergessen: Gedächtnisinhalt zerstört oder nicht auffindbar

Unbewusstes (Unterbewusstsein); Sigmund Freud als Pionier – beim Menschen ist sehr vieles unbewusst gespeichert

soziales Lernen: besondere Sensibilität für Gesichter, Stimmen, Gestik, Mimik, soziale Beziehungen und Hierarchien

die drei wichtigsten Lerntheorien und ihre Grundauffassungen:

Kategorie	Behaviourismus	Kognitivismus	Konstruktivismus
Hirn ist ein	Passiver Behälter	Informationsverarbeitendes "Gerät"	Informationell geschlossenes System
Wissen wird	Abgelagert	Verarbeitet	Konstruiert
Wissen ist	Eine korrekte Input-Output-Relation	Ein adäquater interner Verarbeitungsprozeß	Mit einer Situation operieren zu können
Lernziele	Richtige Antworten	Richtige Methoden zur Antwortfindung	Komplexe Situationen bewältigen
Paradigma	Stimulus-Response	Problemlösung	Konstruktion
Strategie	Lehren	Beobachten und helfen	Kooperieren
Lehrer ist	Autorität	Tutor	Coach, (Spieler) Trainer
Feedback	Extern vorgegeben	Extern modelliert	Intern modelliert
Interaktion	Starr vorgegeben	Dynamisch in Abhängigkeit des externen Lernmodells	Selbstreferentiell, zirkulär, strukturdeterminiert (autonom)
Programmerkmale	Starrer Ablauf, quantitative Zeit- und Antwortstatistik	Dynamisch gesteuerter Ablauf, vorgegebene Problemstellung, Antwortanalyse	Dynamisch, komplex vernetzte Systeme, keine vorgegebene Problemstellung
Software-Paradigma	Lernmaschine	Künstliche Intelligenz	Sozio-technische Umgebungen
"idealer" Softwaretypus	Tutorielle Systeme, Drill & Practice	Adaptive Systeme, ITS	Simulationen, Mikrowelten, Hypermedia

(nach Baumgartner & Payr 1994; WWW)

Erlernen von Regeln in Klassifizierungssystemen

Grundidee: Erfolgreiche Regeln "belohnen" – Stärke der Regel heraufsetzen
Platz in der Regelliste ist begrenzt, Regeln müssen um ihren Platz "kämpfen"

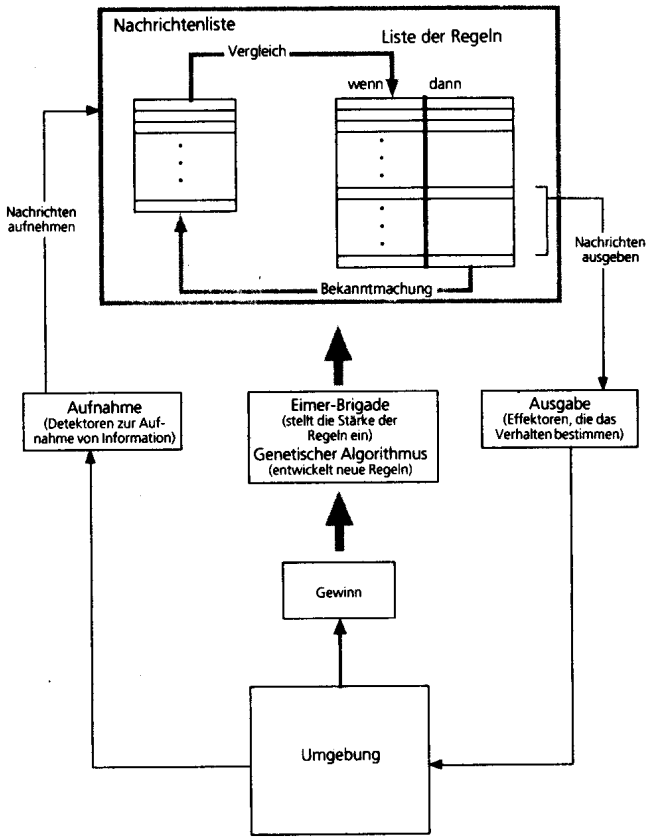
Probleme:

- Regelketten; es würde nur die letzte belohnt
- Erfahrungen treten in keiner festen Reihenfolge auf
- Erfahrungen in der Realität erfordern unterschiedliche (disjunkte oder auch überlappende) Mengen von Regeln

Lösung des Kettenproblems durch "Eimerbrigaden-Algorithmus" (*bucket brigade*):

- jede Regel, die in der Lage war, ihre Nachricht abzugeben, gibt eine "Belohnung" weiter an diejenigen Regeln, die dafür gesorgt haben, dass die Liste in dem passenden Zustand war

→ durch das System geht ein "Stärkefluss", der von Regeln ausgeht, die externe Belohnungen erhalten haben (erfolgreich waren)



Einsatz von lernfähigem Klassifizierungssystem (zusätzlich mit genetischem Algorithmus zum Entwickeln neuer Regeln ausgestattet) in einem künstlichen Tier:

"*Animat*" (animal automat, Begriff von Stewart Wilson)

erster Animat: "*"
einziges Bedürfnis: Nahrung
Umgebung lieferte 92 verschiedene sensorische Eingaben
* wusste anfangs nicht, woran Nahrung zu erkennen war,
es gab Hindernisse ("Bäume" und "Felsen") und eingeschränkte Sehfähigkeit

Wilson sagte nach den ersten Modell-Läufen über *:

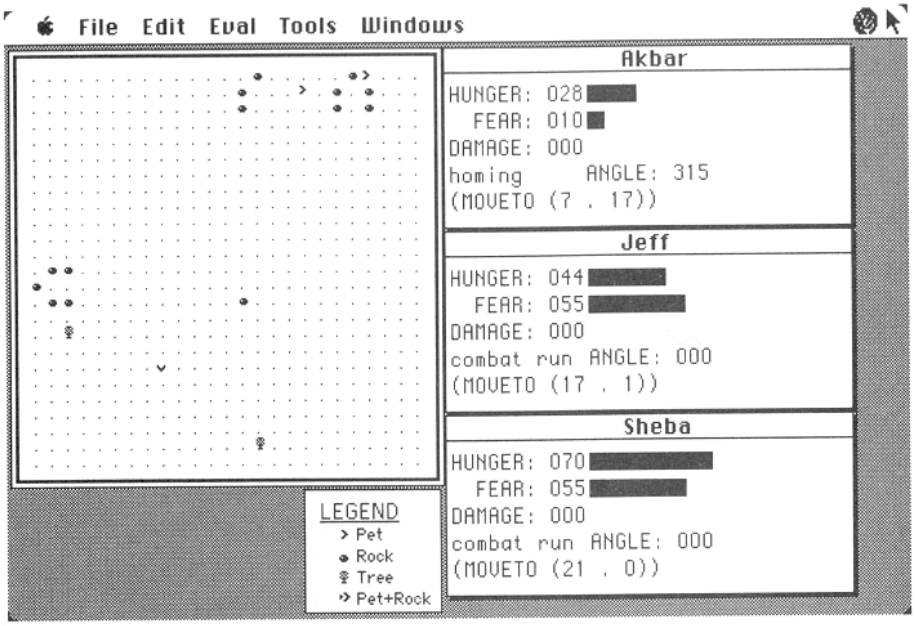
> Er *lernte* tatsächlich. Er schuf Klassifizierungssysteme. Wenn er gegen einen Felsen gelaufen war, erhielt er eine schmerzhafte Rückmeldung, aber da sich häufig Futter in der Nähe der Felsen befand, lernte er nicht nur, nicht gegen den Felsen zu stoßen, sondern auch, nicht einfach wegzulaufen, da sich auf der anderen Seite möglicherweise Futter befand. Und während * sich in dieser Landschaft aufhielt, entwickelt er interessante Verhaltensweisen. Er bewegte sich nicht nach dem Zufallsprinzip, sondern ließ sich normalerweise in eine Richtung treiben, was übrigens auch die beste Taktik für jemanden ist, der sich im Nebel verlaufen hat. Sobald er etwas sah, reagierte er augenblicklich. Wenn es Futter war, fraß er das Futter auf der Stelle, und wenn er auf einen Felsen stieß, bewegte er sich um ihn herum.

Variante: hierarchische Organisation von ganzen "Verhaltens-Modulen" (nicht bloß Tabelle einzelner Regeln)

– Beispiel: *Petworld* (Bill Coderre, MIT)

Petworld:

- 2-dim. Welt mit Tierchen, Bäumen (die die Tiere nach Nahrung absuchen), Steinen
- Tierchen mit Blickfeld von 90°
- können einzelne Steine tragen
- Tierchen begegnen sich als Feinde (keine Fortpflanzung)
- interne Zustände (Hunger, Verletzung, Angst, Nestlokalisierung, Nutzlast) mit Wertebereichen zwischen 0 und 100
- Gehirn besteht aus Hierarchie von Modulen ("Experten"), jedes ist auf eine Verhaltensweise spezialisiert (z.B. Nahrungssuche, Nestbau)
- Hierarchiestufe hängt von der Bedeutung des Verhaltens ab
- jedes Modul bekommt Eingaben von untergeordneten Modulen und von der Umwelt
- als Reaktion wird eine Rangliste möglicher Verhaltensweisen zur Verfügung gestellt und nach oben weitergereicht
- Ranglisten werden fortschreitend modifiziert (Ergebnis sind Kompromisse, z.B. zwischen Kämpfen und Nahrungssuche, situationsabhängig)
- die vom höchstrangigen Modul empfohlene Aktion wird ausgeführt – und gespeichert (Erinnerung)
- Handlungsempfehlungen können nach dem Erfolg bewertet werden, dadurch Lernfähigkeit (optional)

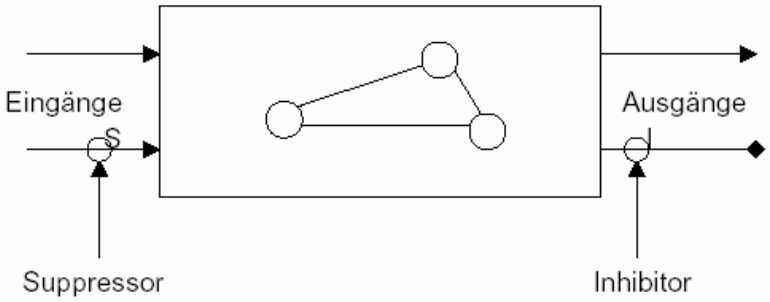


Beobachtung: Neubildung von Verhaltensweisen, z.B. lagerte ein Tierchen zusätzliche Steine in der Nähe seines Nestes

Prinzip der Verhaltens-Module ähnelt der *subsumierenden Architektur* bei Robotern (Rodney Brooks):

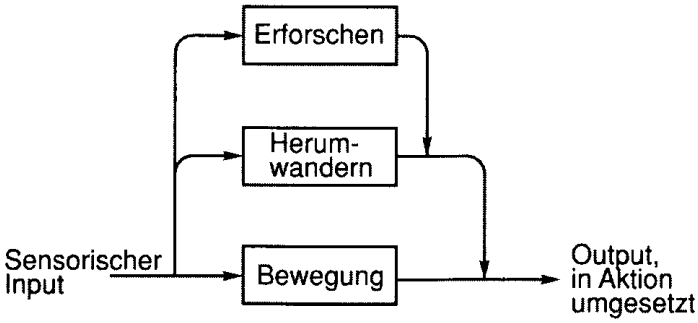
Subsumptionsarchitektur

- häufig in autonomen Robotern eingesetzt
- Idee der verhaltensbasierten Aufteilung
- Konkrete Architektur
- Agent besteht aus einzelnen Modulen mit endlichen Automaten
- Verbindungen zwischen Modulen können verändert werden (Suppressor und Inhibitor)



- Hierarchische Struktur der unabhängigen Module (über Suppressor und Inhibitor)
  - Kollisionsvermeidung über Herumwandern
  - Kollisionsvermeidung über Wegplanung
  - Wegplanung über Herumwandern

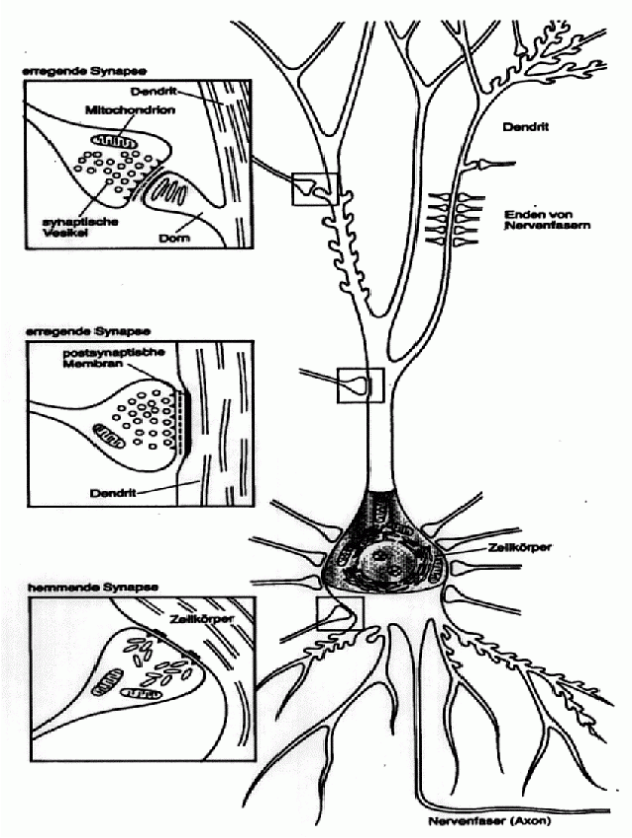
(Klügl 1998)



(Thro 1994)

wie funktioniert Lernen im "natürlichen" Leben?

- nicht vollständig verstanden
- Neubildung, Inhibition und Verstärkung von Verschaltungen zwischen Neuronen spielt wichtige Rolle
- auch chemische Prozesse beteiligt
- extremer Stress in frühen Phasen des Gehirnwachstums kann langfristig hormonale Konzentrationen verändern und die kognitiven Leistungen einschränken



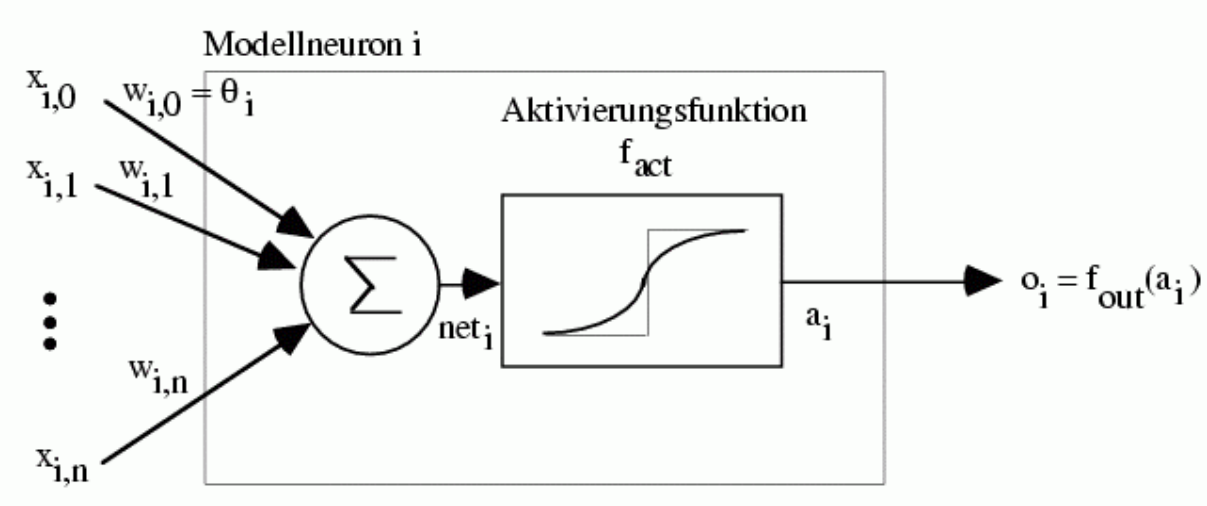
Nervenzelle (Neuron)
(aus Levi 2002)

Ansatz der Nachbildung des zentralen Nervensystems der höheren Tiere, bzw. des Gehirns:
"Konnektionismus" (schon früh ein Ansatz der AI-Forschung, dann zeitweise "aus der Mode" gewesen) –

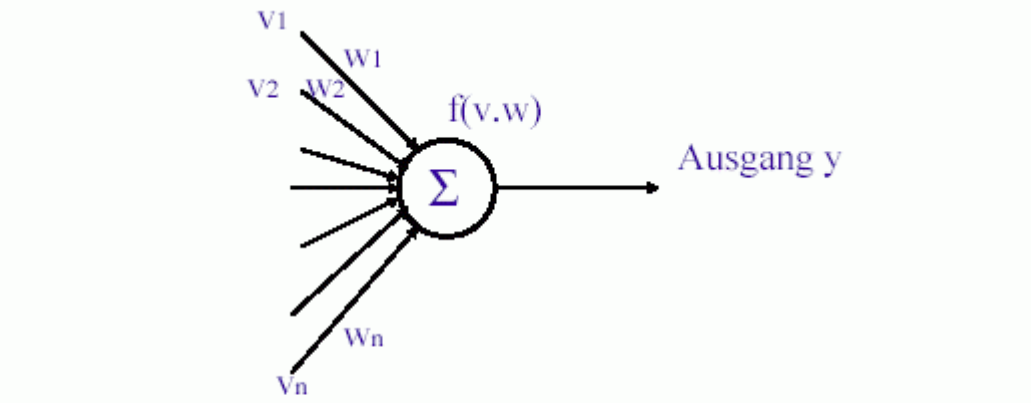
künstliche neuronale Netze (kurz knN oder NN)

Modellneuron:

Dendriten und Synapsen $\rightarrow$ gewichtete Verbindungen w_{ij}
Hemmung (Inhibition) $\rightarrow$ $w_{ij} < 0$
Erregung (Exzitation) $\rightarrow$ $w_{ij} > 0$
keine Verknüpfung $\rightarrow$ $w_{ij} = 0$
Zellkörper $\rightarrow$ Aktivierungszustand a_i, Aktivierungsfunktion f_{act}
Axon $\rightarrow$ Ausgabefunktion f_{out}



vereinfacht:



Gesamteingang:

$$x = \sum_{i=1}^{n} v_i w_i \qquad x = \sum_{i=1}^{n} v_i w_i - \theta$$

Transferfunktion:

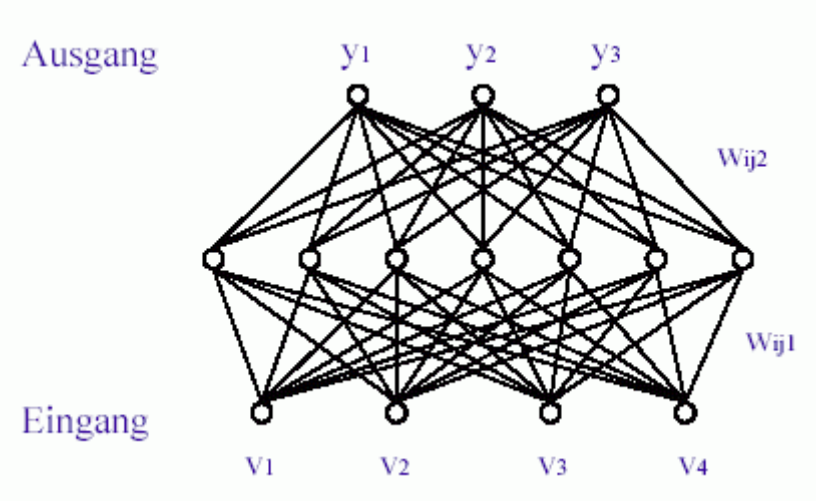
$$f(x) = \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0 \end{cases} \qquad f(x) = \frac{1}{1+e^{-x}}$$

(Beispiele)

Neuronales Netz:

- Zusammenschaltung mehrerer Neuronen
- Ausgang eines Neurons wird Eingang eines anderen Neurons
- imitiert die hohe Anzahl von Verbindungen einfacher Neuronen im menschlichen Hirn
- Netzwerk besteht aus mehreren Eingängen und mehreren Ausgängen

Beispiel: Feed-Forward-Network mit 1 hidden layer:



Einteilung der Schichten in

- Eingabeschicht
- verdeckte Schichten (*hidden layers*) (optional)
- Ausgabeschicht.

Achtung: in der Literatur wird bei der Zählung der Schichten die Eingabeschicht oft nicht mitgezählt.

Lernverfahren für knN

(a) überwachtes Lernen (*supervised learning*), "Lernen mit Lehrer":

- Trainingsmenge von Eingabe- und Ausgabemustern
- zu jedem Eingabemuster existiert ein eindeutiges korrektes (bestes) Ausgabemuster

Lernen:
Die Gewichte und evtl. Schwellenwerte werden solange durch nochmaliges Anlegen der Eingabemuster verändert, bis die Paarung (Eingabemuster, Ausgabemuster) für die Trainingsmenge stimmt.

Generalisierung:
Ähnliche Eingabemuster, die nicht zur Trainingsmenge gehören, werden nach der Trainingsphase entweder in bereits trainierte Ausgabemuster (Perzeptron) oder in ähnliche Ausgabemuster (Backpropagation-Netzwerke) überführt.

(b) unüberwachtes Lernen (*unsupervised* / *self-organized learning*):
Lernen erfolgt durch Selbstorganisation. Ähnliche Eingabemuster werden assoziativ als ähnlich klassifiziert.
Beisp.: Kohonen-Netze.

Lernregeln für die Neuronen:

Hebb'sche Regel (Donald Hebb, Neurophysiologe, 1949):

"Wenn Neuron *i* und Neuron *j* zur gleichen Zeit stark aktiviert sind, dann erhöhe das Gewicht w_{ij}, das diese beiden Neuronen verbindet"
in Formeln:

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta w_{ij}$$

$$\Delta w_{ij} = \alpha y_i y_j$$

(y_i : Ausgabewert von Neuron i)

Motivation: Beobachtungen an "echten" neuronalen Netzen

Nachteil: Gewichte können nur größer werden $\Rightarrow$ zu wenig flexibel.

Variante für einschichtige Netze: statt der Ausgangswerte werden Eingangswert und Sollwert eingesetzt

$$\Delta w_{ij} = \alpha \omega_i E_j$$

α heißt "Lernrate".

Delta-Lernregel (Widrow-Hoff-Regel):

$$\Delta w_{ij} = \alpha(\omega_j - y_j) y_i$$

(ω_j = Sollwert für Neuron *j*)

- Gewichte können größer und kleiner werden
- Gewichtsänderung ist proportional zum Fehler an den Ausgabe-Neuronen
- in dieser Form nur für Feedforward-Netzwerke mit 2 Schichten sinnvoll (Perzeptron)

- Für Perzeptron kann Erfolg der Lernregel garantiert werden
- aber nur eingeschränkte Mächtigkeit: nur Teilmenge der möglichen (Eingabe, Ausgabe)-Funktionen kann gelernt werden

Verallgemeinerung der Delta-Regel für Netzwerke mit mehr als 2 Schichten (d.h. mit hidden layers):

Training durch Backpropagation (Fehlerrückführungs-Netz); Fehlerfunktion an den Ausgabeneuronen wird minimiert

Künstliche neuronale Netzwerke werden z.B. in den "Creatures" von Karl Sims und in dem gleichnamigen Computerspiel eingesetzt.

Architektur der knN: meist vorgegeben ("genetisch"), Gewichte werden durch Lernen modifiziert.

Beispiel: David Ackley & Michael Littman: "AL"

- 100 × 100 - Gitter
- *Agenten*
- Pflanzen (Futter)
- Bäume
- Felsen (Wände)
- Fleischfresser (Feinde)

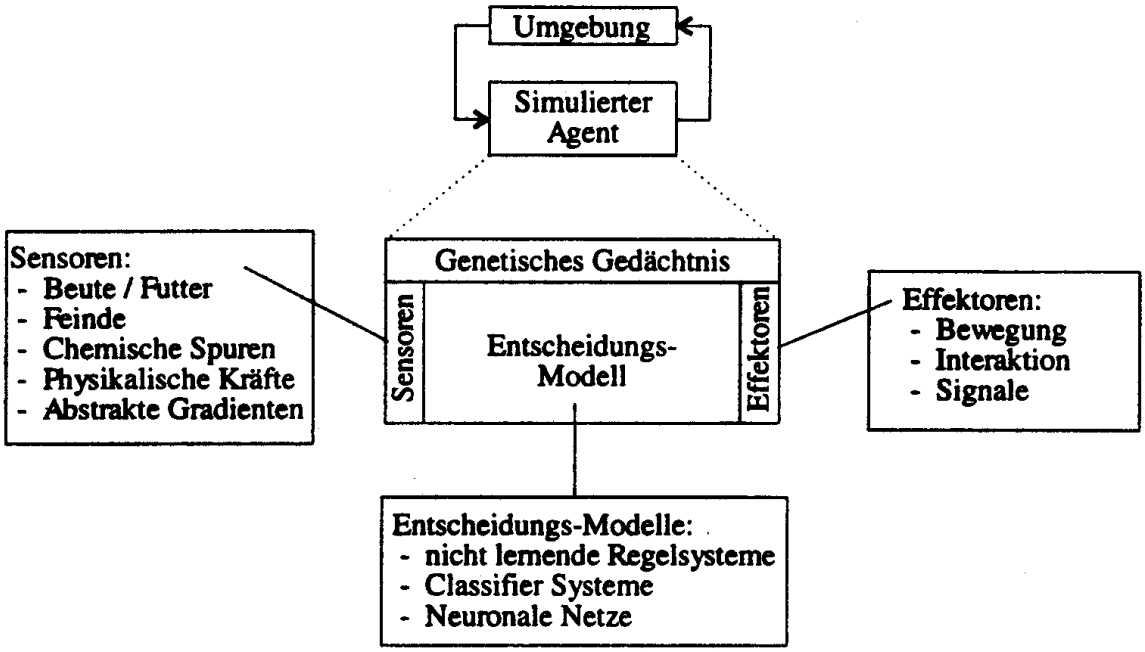
- jeder Agent kennt den Inhalt von 4 Nachbarzellen in jeder Richtung
- jeder Fleischfresser 6 Nachbarzellen in jeder Richtung

Gehirn der Agenten: ein Auswertungs- und ein Aktionsnetzwerk

- *Auswertungs-Netzwerk:* übersetzt sensorische Informationen in eine numerische Rangordnung
- *Aktions-Netzwerk:* setzt auf der Basis der Ziele aus dem Auswertungs-Netzwerk Informationen in Verhaltensweisen um, misst den momentanen Erfolg eines Agenten, vergleicht ihn mit früheren Auswertungen und ermöglicht Lernen durch Verstärkung!

- sexuelle Fortpflanzung
- Vererbung (siehe später)

Zusammenfassung zur Verhaltenskontrolle in Animaten:



(aus Bartscht & Müller-Schloer 1995)

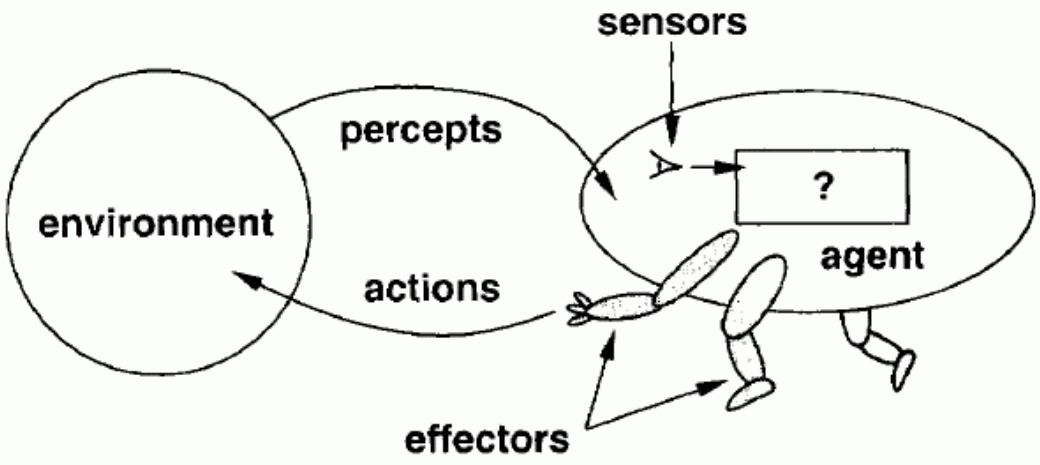
Kritik an Animaten-Welten wie Petworld, "AL", Creatures etc.:

- Begrenztheit dieser Welten
- "*Pacman-Syndrom*": die Szenarien sind letztlich "gestellt", haben sich nicht natürlich entwickelt (z.B. Art der Feinde, Wände etc....) – es bleibt ein Spielcharakter
- Kritik an einzelnen Konstruktionsprinzipien, z.B. sind Ethologen (Verhaltensforscher) von hierarchischen Verhaltensmodellen wieder abgekommen

Das Konzept der "Agenten" findet jedoch in den letzten Jahren immer mehr Interesse – auch zum praktischen Problemlösen

„Agent“

„Asking the question of what an agent is to a DAI researcher is as embarrassing as the question of what intelligence means is to an AI researcher”
(C. Hewitt, DAI-Workshop, 1994)



(Klügl 1998; DAI = distributed artificial intelligence)

Russell & Norvig 1995:

"An agent is anything that can be viewed as perceiving its environment through its sensors and acting upon that environment through effectors"

Eigenschaften eines Agenten:

- er existiert (mit gewisser Dauer) in seiner Umgebung (*Situatedness* und *Permanenz*)
- *autonom*: Verhalten wird durch den Agenten selbst bestimmt, ohne Kontrolle von außen
- *reaktiv*: Agent kann Ereignisse wahrnehmen und auf dieser Grundlage seine Aktionen abstimmen (*Responsiveness*)
- *proaktiv*: Agent reagiert nicht nur auf Reize, sondern kann von sich aus die Initiative ergreifen
- *sozial*: strukturierte Kommunikation mit anderen Agenten ist möglich ("social abilities")

weitere Eigenschaften:

- mentale Konzepte (Wissensverarbeitung, Affekte, Ziele)
- Ziel-Orientiertheit
- "Rationalität"
- Mobilität
- Adaptivität
- "Aufrichtigkeit", "Gutwilligkeit" (?)

Agent versus Objekt

- Objekt kapselt seinen Zustand und stellt mit Methoden Schnittstellen nach außen zur Verfügung, mit denen Zustand verändert werden kann. Ein Objekt hat aber keine Kontrolle, wann diese Methoden aufgerufen werden.
- Agent kapselt Verhalten: Über die Schnittstellen nach außen (Sensoren) kommen Anfragen. Der Agent entscheidet, ob und wie er diese Anfragen bearbeitet.
- „Objects do it for free, agents for money“

Agentenarchitekturen

Taxonomie nach Genesereth/Nilson 1987

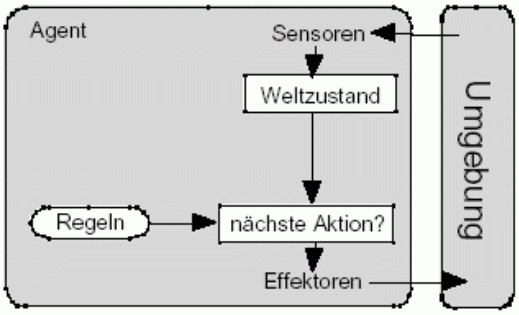
- „Tropistic agent“: ⟨S,T,A,see,do,act⟩ mit
  S: Menge aller externen Zustände
  T: Menge aller Teilmengen von S, mit Elementen, die Agent nicht unterscheiden kann (bestimmt durch sensorischen Fähigkeiten)
  see: S→T Sensorfunktion
  A: Menge aller Aktionen, die Agent ausführen kann
  do: A×S → S
  act: T → A
- „Hysteretic agent“: ⟨I,S,T,A,see,do,internal,act⟩
  ist ein tropistic agent mit internem Zustand:
  I: Menge aller für den Agenten unterscheidbaren internen Zustände.
  Speicheraktualisierung: internal:I×T → I
  act: I×T → A
- „Knowledge level agent“:
  ⟨D,S,T,A,see,do,database,act⟩
  Abstraktere Beschreibung eines intelligenten Agenten: Interner Zustand ist eine Datenbank von Fakten, mit denen der Agent sein Wissen über die Umwelt repräsentiert.
- „Stepped knowledge level agent“

Literatur:

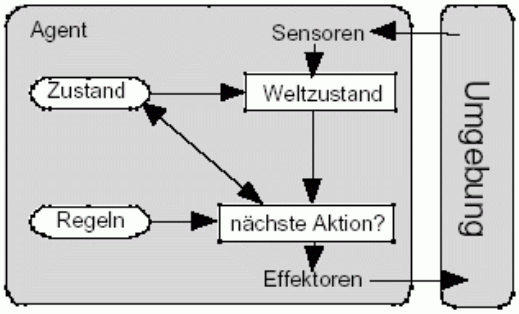
M. Genesereth & N. Nilson (1987) Logical Foundations of Artificial Intelligence, Morgan Kaufmann

Architektur-Taxonomie nach Russell/Norvig

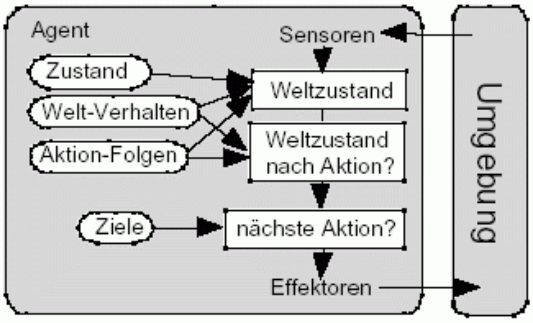
Einfacher Reflex-Agent



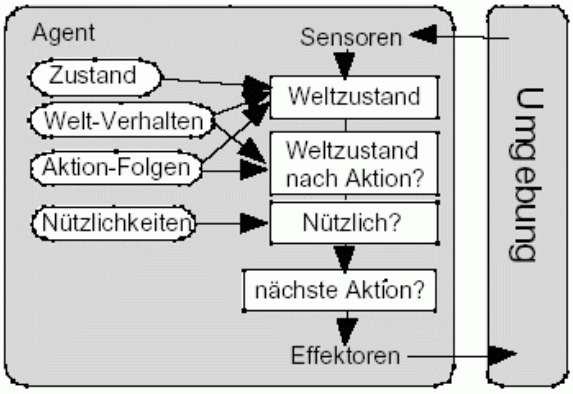
Reflex-Agent mit internem Zustand



Agent mit expliziten Zielen



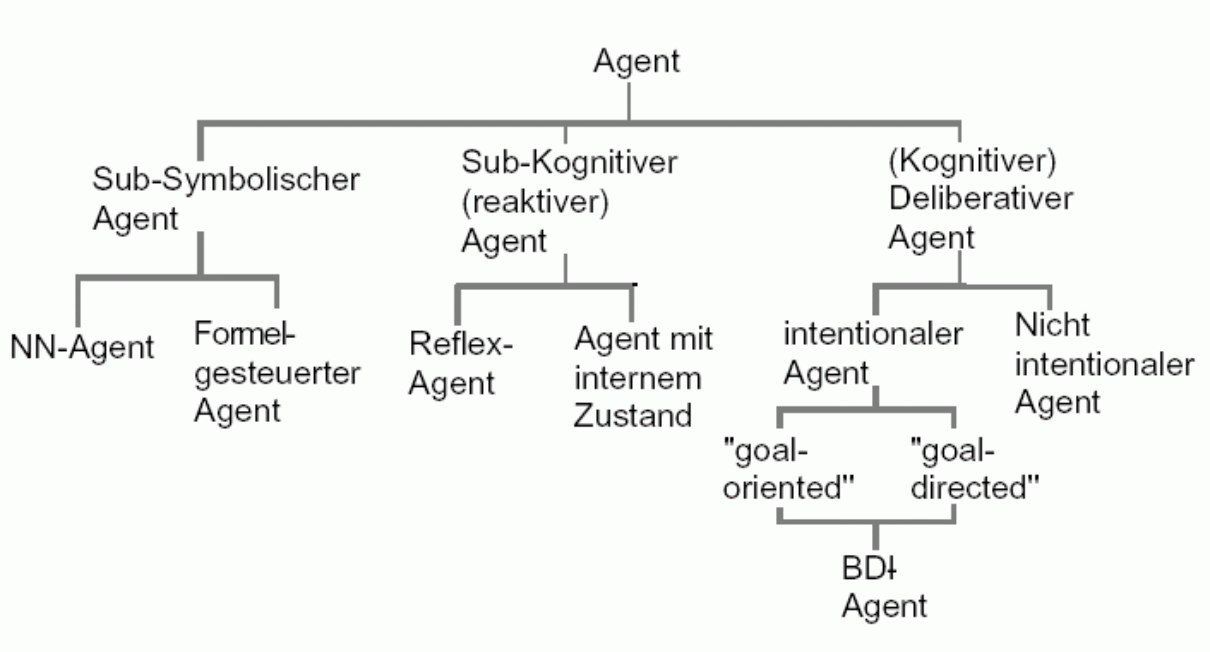
Utilitaristischer Agent



Literatur:

S. Russell & P. Norvig: Artificial Intelligence - a Modern Approach, Prentice Hall, 1995

Klassische Agentenarchitekturen (nach Klügl 1998):



(BDI = beliefs - desires - intentions)

Subkognitive Agenten:
direkte Verknüpfung zwischen Sensorik und Effektoren

- Reflex-Agenten
- Tropistische Agenten (vgl. L-System-Modell der pflanzenfressenden Tiere)
- Stimulus-Response-Systeme

meist ohne inneren Zustand, oder nur mit sehr einfachen Zustandsmengen

"Zustands-Aktions-Abbildungen"
Alle in einer Situation sinnvollen Pläne werden kompiliert und in Situation-Aktion-Regeln gespeichert
Beispiele:

- Klassifizierungssysteme (einfache)
- "Universal Plans"
- Teleo-Reactive Programs (T-R)
- Situated Control Rules (SCR)
- Markoffketten-Entscheidungsmodelle (Markov Decision Models, MDM)

Ausgehend von jedem mögl. Zielzustand wird Aktion bestimmt, die Nähe zu Zielzustand minimiert
starke Vorauss. an Modellierung der Umgebung; wenig effizient

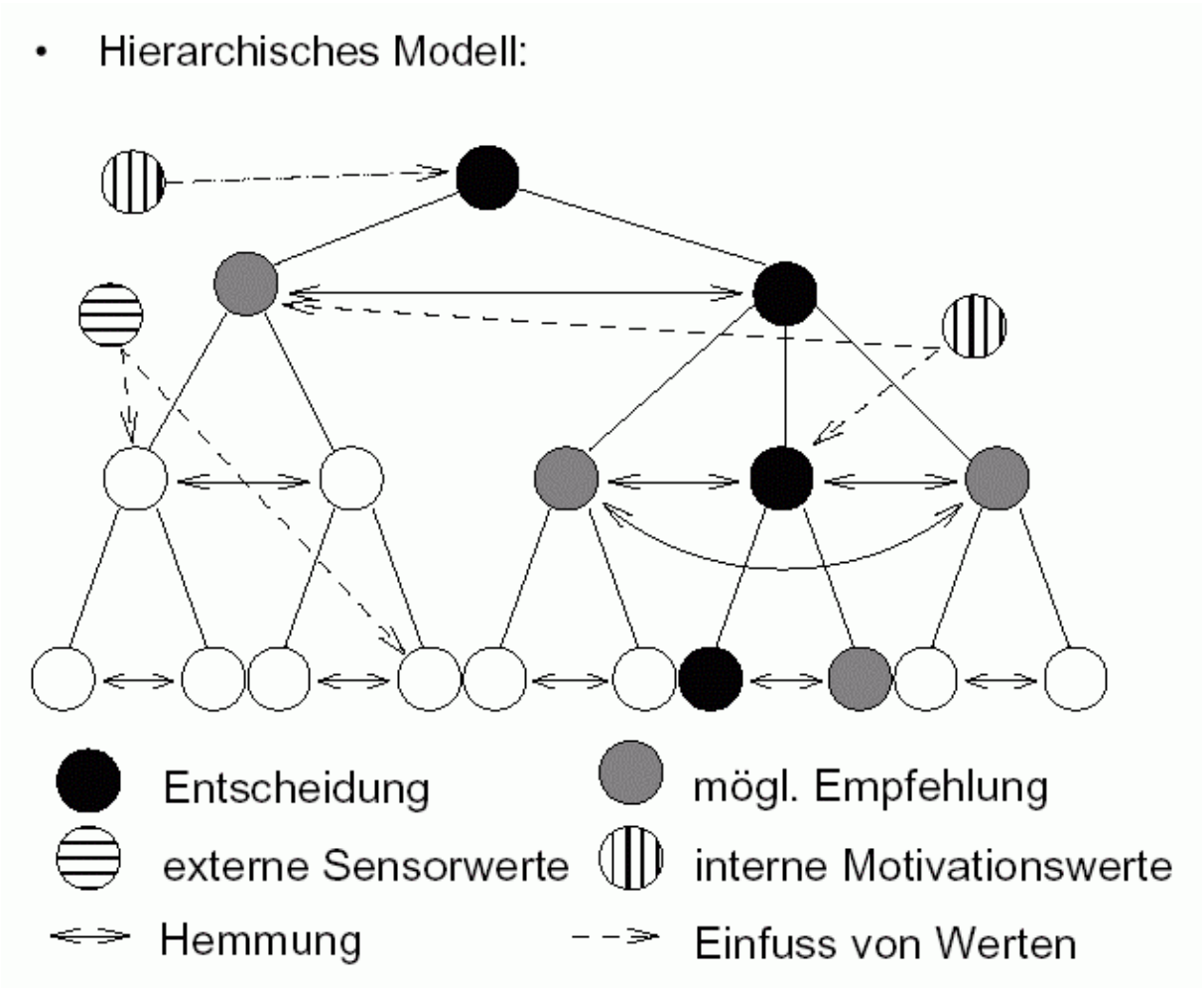
Agenten-Architekturen mit Netzstruktur:

Aktionsselektion wird durch Interaktion einzelner Strukturen eines Netzes erreicht
→ Emergenz innerhalb des Agenten

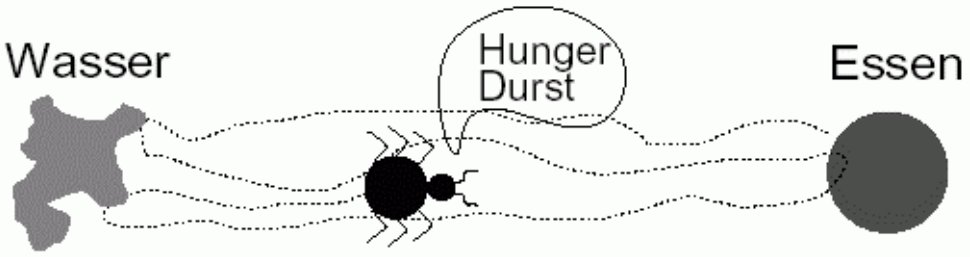
Beispiele:

- knN
- Subsumptionsarchitektur
- Maes'sches Netzwerk
- Ethologische Modelle

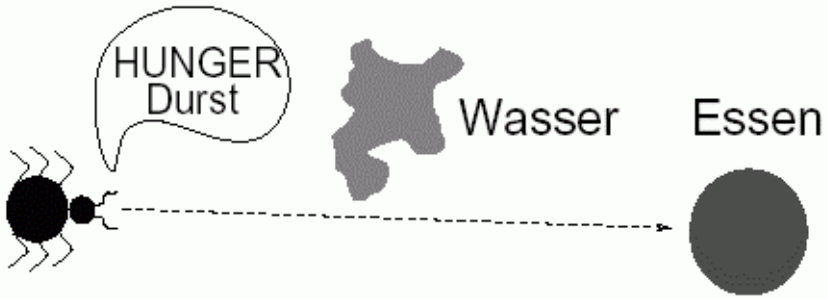
Beispiel: Ethologisch inspiriertes Modell von Blumberg (1994)



- Zeitliche Aspekte von Verhalten:
  - Verhalten (z.B. Essen) von unterschiedlicher notwendiger Dauer
  - Kein Verhalten ‚vernachlässigen'



- Opportunismen:
  Ausnutzen günstiger Gelegenheiten



- Kombination externer Sensorwerte (keine Prädikate) und interner Motivationswerte

Kognitive Agenten

- Kognitive Architekturen basieren auf kognitiven Prozesse, d.h. auf Prozessen, die Aufnahme, Speicherung und Gebrauch von Wissen involvieren, das nicht direkt an Verhalten gebunden ist. (Toates, 1994)
- Notwendig ist eine interne Repräsentation dieses Wissens
- Extrapolation über aktuelle Sensordaten hinaus möglich.
- ≈ Deliberative, meist intentionale Agenten
- Ziele beim Einsatz von kognitiven Architekturen
  - zur Problemlösung durch intelligenten Agenten-Steuerung
  - Konstruktion *autonomer* Agenten
  - als Modelle von Kognition
  - Konstruktion interessanter Interaktionspartner für Menschen
- Weitere Unterteilung:
  - Allgemeine Architekturen, z.B. Soar
  - Planer-Architekturen, z.B. reaktive Planer, wie RAP
  - BDI - Architekturen, wie IRMA, PRS oder Tok-Architektur
  - Hybride Architekturen, wie InteRRap

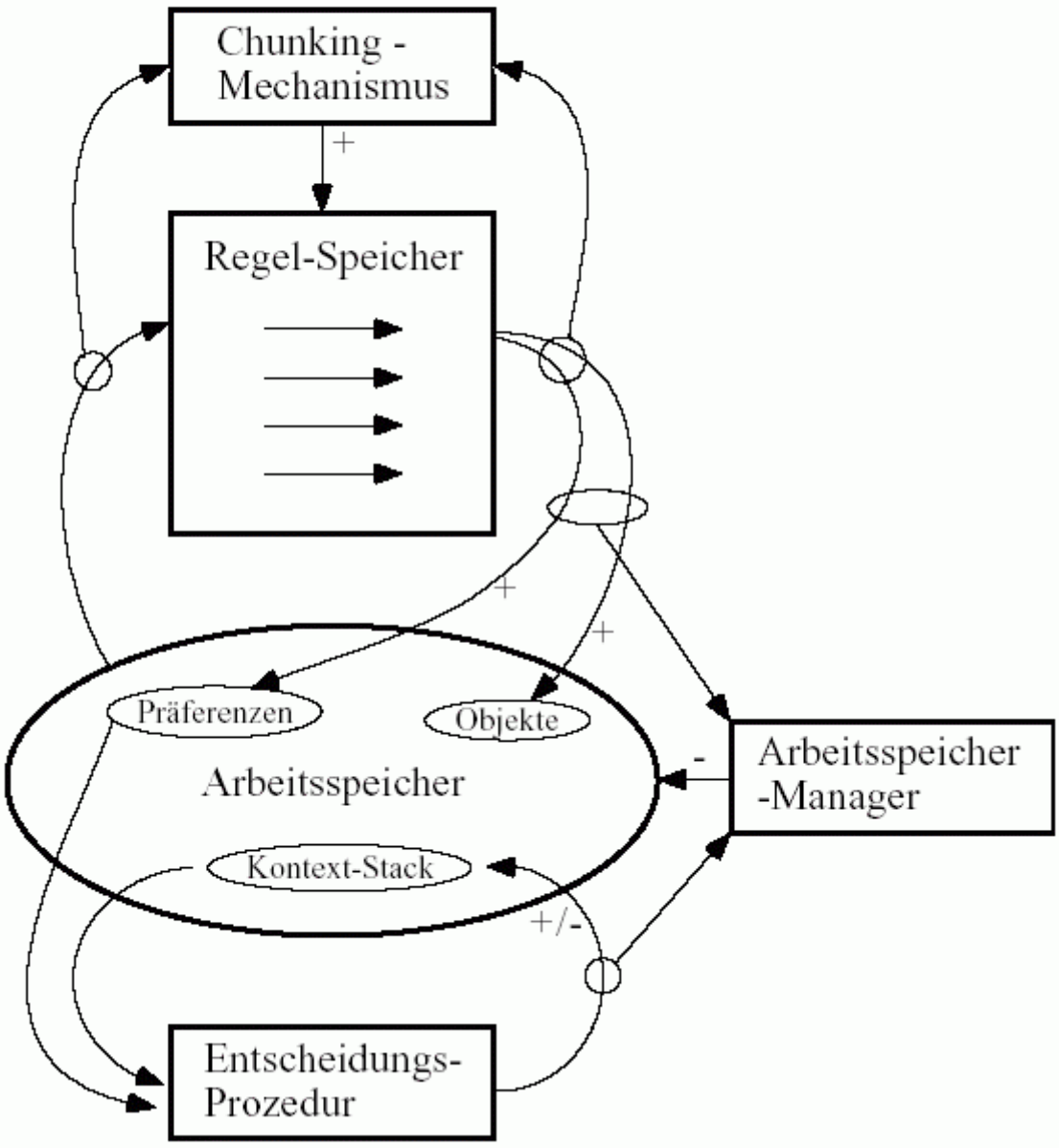
Allgemeine Architekturen, z.B. Soar

- Soar : „An architecture for a general intelligence“
  Universell einsetzbare, einheitliche Architektur
- „problem space hypothesis”: Alles Problemlösen besteht aus einer (heuristischen) Suche in einem Problemraum
- Bestandteile
  - langfristiges Wissen in Form von Regeln:
    “Task-Implementation”- und “Such-Kontroll”-Regeln
  - Aktionen der Regeln erzeugen Elemente des Arbeitsspeichers als kurzfristiges Wissen:
    Problem-Räume, Zustände, Operatoren, Präferenzen
  - Kontext-Stack mit den aktuell verfolgten Zielen
- Processing in zwei Phasen:
  - „Elaboration-Phase“: Regeln aus Regel-Speicher feuern, bis keine mehr Feuern kann ➜ Hinzufügen von Arbeitsspeicher-Elementen
  - „Decision-Phase“: Auf Basis der vorhanden Präferenzen wird Problemraum, Situation oder Operator ausgewählt.
    Bei Sackgasse automatisches Erzeugen eines neuen Ziels (Kontext)
- Chunking-Mechanismus lernt „Abkürzungen“ für Sackgassen

Literatur:

J. E. Laird, A. Newell & P. S. Rosenbloom (1987): *Soar: An Architecture of a general intelligence*. In. Artificial Intelligence, 33:1-64, 1987

Soar-Architektur



RAP

„Reactive Action Packages“

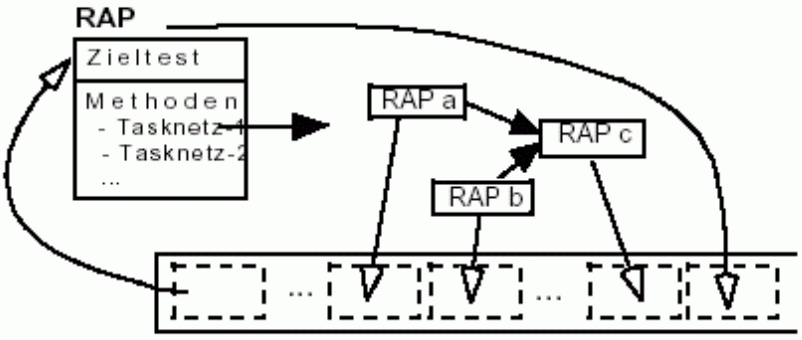
RAP

Kontext
Zieltest
Methoden - Tasknetz-1 - Tasknetz-2 ...

Reaktive Aktionspakete

- Vorgehensweisen, um das in *Zieltest* angegebene Ziel zu erreichen, werden in *Tasknetzen* spezifiziert.
- Hierarchische Strukturierung durch Verweise in Tasknetzen auf andere RAPs möglich.
- *Kontext* zum Test der Ausführbarkeit in der aktuellen Situation

Vorgehensweise



Wähle RAP aus Task-Agenda

Wenn primitives RAP ➜ führe Aktionen aus
wenn dabei gescheitert, entferne RAP und alle der selben Methode.

Wenn normales RAP
Zieltest ➜ Erfolg, RAP aus Agenda entfernen
sonst Task-Netz auswählen, in Agenda einsortieren und erstes RAP an das Ende der Task-Agenda setzen

Probleme: Kein Erkennen von Opportunismen, Schleifengefahr

Beispiel

Bewegen eines Objekts mit einem Verteil-Roboter

```
(DEFINE-RAP
      (INDEX (move-to ?thing ?place))
      (SUCCEED (location ?thing ?place))
      (METHOD
            (CONTEXT (and (location ?thing ?loc) (not (= ?loc unknown))))
            (TASK-NET
                  (to (goto ?loc)
                        ((truck-location ?loc) for t1))
                  (t1 (pickup ?thing)
                        ((truck-holding ?thing) for t2)
                        ((truck-holding ?thing) for t3))
                  (t2 (goto ?place)
                        ((truck-location ?place) for t3))
                  (t3 (put-down ?thing))))
      (METHOD
            (CONTEXT (and (location ?thing unknown)
                              (not (truck-location warehouse))))
            (TASK-NET
                  (t0 (goto warehouse)
                        ((truck-location warehouse) for t1))
                  (t1 (pickup ?thing)
                        ((truck-holding ?thing) for t2)
                        ((truck-holding ?thing) for t3))
                  (t2 (goto ?place)
                        ((truck-location ?place) for t3))
                  (t3 (put-down ?thing)))))
```

Wenn keine der beiden Methoden anwendbar ist, schlägt das RAP fehl.

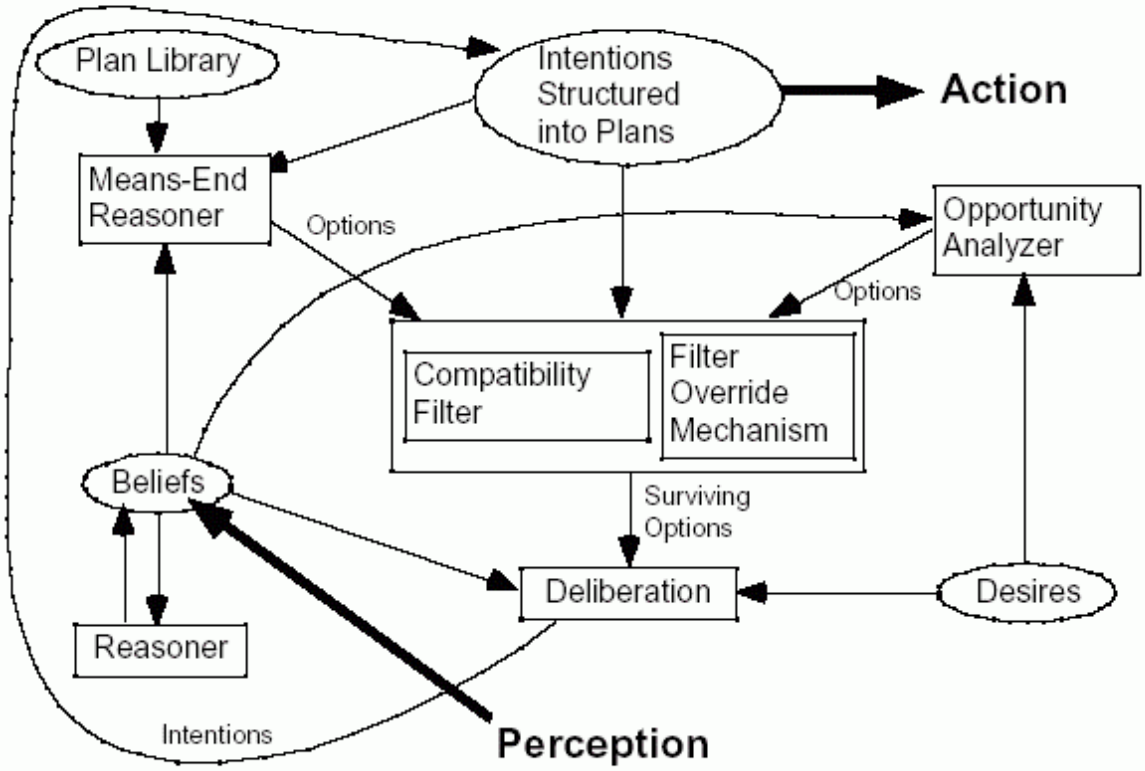
BDI-Architekturen (Beliefs - Desires - Intentions)

- Intentionalität beinhaltet drei „mentale Konzepte“ (aus „folk psychology“)
  - Kognitive
    „Beliefs“ - Wissen über Umwelt, Datenbasis
  - Affektive
    „Desires“ - Motivationen und grobe Ziele des Agenten
  - Konative
    „Intentions“ - konkrete Ziele und Pläne, mit denen diese erreicht werden können, Verpflichtungen
- Agenten als „Intentionale Systeme“:
  - Abstraktion in mentalen Konzepten als nützliche Beschreibungsstruktur und Verhaltenserklärung
  - Implementierungs-unabhängig
  - Als Reasoning-Grundlage für „introspective agents“ (Schließen über eigene Beliefs, Desires, Intentions) oder für soziale Agenten, die so das Verhalten anderer Agenten repräsentieren.
- Beispiele:
  IRMA, PRS als erste, grundlegende BDI-Architekturen
  COSY (Burmeister & Sundermeyer, 1992)
  GRATE* (Jennings, 1993)

IRMA

„intelligent resource-bounded machine architecture“

- Planen mit beschränkten Ressourcen (Zeit und Wissen)



- Rechtecke als Prozesse, Kreise als Informationsspeicher
- Unterscheidung zwischen „plans as recipes“ (Wissen, welcher Plan ist wann wofür anwendbar) im *Plan Library* und aktuell ausführbaren Plänen (im *Intentions structured into Plans*)
- Basis: „A rational agent is committed to what she plans“
- Tests im Tileworld-Szenario

Literatur:

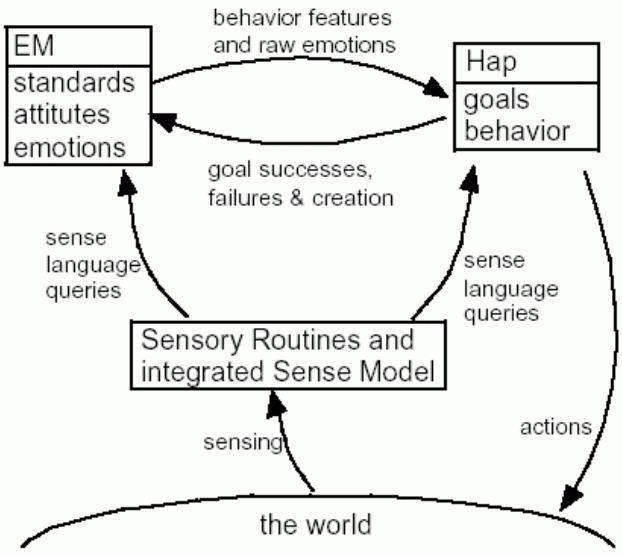
M. Bratman, D. I. Israel, M. E. Pollack (1988): *Plans and resource-bounded practical reasoning.* In. Computational Intelligence 4:349-355

Bestandteile der IRMA-Architektur:

- Vorschläge für Optionen (Pläne) kommen von:
  - *means-end reasoner* realisiert eine Mittel-Ziel-Analyse und schlägt Teilpläne zu etablierten Intentionen vor.
  - *opportunity analyzer* untersucht Umweltveränderung und weist auf Gelegenheiten zur Erfüllung von „desires“
- Mögliche Optionen werden durch *compatibility filter* überprüft auf Kompatibilität mit aktuellen Intentionen. Bei Inkompatibilität überprüft *filter override mechanism* die Optionen bzgl. Wichtigkeit relativ zu den Intentionen.
- „überlebende“ Optionen werden im Deliberation-Prozess gegeneinander abgewägt (unter Berücksichtigung von beliefs und desires) ➔ Update der *Intentions structured into plan* ➔ Ausführung der Aktionen

Bates et al. (1992): "Tok-Architektur"

- Oz - Projekt
- „believable agent“:
  Architektur für „Virtual Reality“ - Agent, der möglichst realistisches Verhalten aufzeigen soll: Hauskatze „Lyotard“



Literatur:

J. Bates, A. B. Loyall and W. S. Reilly (1992): An Architecture for Action, Emotion and Social Behaviour. In. C. Castelfranchi and E. Werner (eds): Artificial Social Systems (MAAMAW'92), Springer, pp. 108-117.

Zur Tok-Architektur

- Simulierte Umwelt: diskret, geometrisch mit Repräsentation räumlicher Relationen („auf dem Tisch“)
- Weltmodell wird sukzessive aufgebaut. Repräsentation in Form eines Graphen aus Objekten der Umwelt und ihren Verbindungen. Daten mit Zeitstempel
- Hap-Modul zur Bestimmung der nächsten Aktion
  Aktive Ziele in Und-Oder-Baum organisiert, mit jedem Ziel fest ein oder mehrere Pläne verknüpft. Generierung von Zielen über „behavioral features“.
- Em-Modul für Generierung und Aktualisierung von Emotionen und soziale Bindungen des Agenten.
  - Emotionen werden über der Vergleich von Ziel-Erfüllbarkeit, Erfüllung von Zielen mit externe Ereignissen gesteuert
  - „Standards“ bilden Beurteilungsgrundlage für eigene und fremde Aktionen („hilft-mir-meine-Ziele-zu-erreichen“).
  - „Attitudes“ sind Bewertungen, die Objekten der Umwelt gegenüber aufgebaut werden.

Emotionen, etc. von Lyotard

Emotionen
- Angst
- Freude
- Traurigkeit
- Bewunderung
- Vorwurf
- Dankbarkeit
- Haß
- Liebe
- Wut
- (Hoffnung, Stolz, Scham, Befriedigung)

„behavioral features“
- Neugierig
- Aggressiv
- Ignorierend
- Freundlich
- Energisch
- Zufrieden
- (stolz)

Verhalten
- Sich streicheln lassen wollen
- Sich säubern
- Raus/rein wollen
- Essen wollen
- Ein Objekt bekommen
- Nach etwas suchen
- Mit einem Ball spielen
- Mit der Maus spielen
- Verstecken
- Objekte bewegen
- Schnurren
- Buckel machen
- Zischen
- Töten
- Beißen
- Davon rennen
- Spaß haben
- Ball jagen
- Lecken
- Sich an etwas reiben
- Etwas anschauen
- An einem sonnigen Platz liegen