SIEMENS

CORPORATE TECHNOLOGY

Vorlesung

Multiagentensysteme

Dr. Jörg Müller
Siemens AG, CT IC 6
joerg.p.mueller@siemens.com

SIEMENS

CORPORATE TECHNOLOGY

Vorlesung Autonome Intelligente Systeme

2. AGENTENARCHITEKTUREN, WISSENSREPRÄSENTATION UND REASONING

SIEMENS

CORPORATE TECHNOLOGY

Informelle Definitionen: Agentenarchitektur

'[A] particular methodology for building [agents]. It specifies how . . . the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions . . . and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology.' **[Pattie Maes]**

'*[A] specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.'* [Leslie Kaelbling]

SIEMENS

CORPORATE TECHNOLOGY

Historische Entwicklung

- **18. Jhdt.: Regelung von Dampfmaschinen durch mechanisches Feedback (J. Watt)**
- **Ca. 1930: Kybernetik — mathematische Modelle für analoge Regelung**
- **Ca. 1950: Physical symbol systems hypothesis (Simon and Newell)**
  Eine Voraussetzung für intelligentes Verhalten ist die Fähigkeit, Aspekte der Welt symbolisch zu repräsentieren. (= Verwendung logischen Schließens für die Aktionsauswahl, "starke KI")
- **Prinzipielle Probleme des Symbolischen Ansatzes führen zum Paradigma Reaktiver Agenten (seit ca. 1985)**
- **Seit ca. 1990, pragmatische Ansätze, hybride Architekturen, Schichtenarchitekturen**

Arten von Agentenarchitekturen

- Symbolische / logische Architekturen: Deliberative Agenten
- Architekturen für „Praktisches Schließen“
- Reaktive Architekturen
- Hybride oder Mehrschichten-Architekturen

3.1 Deliberative Agenten

- Basieren auf Prinzipien der symbolischen Künstlichen Intelligenz
- Werden als wissensbasierte Systeme gesehen
- Besitzen interne und symbolische Repräsentation ihrer Umgebung (= Wissen)
- Treffen Entscheidungen auf der Basis dieses symbolischen “mentalen Zustandes” durch symbolische Reasoning-Prozesse (Reasoning=Schließen).

Hauptprobleme beim Design deliberativer Agenten

- **Abbildungsproblem:**
  - **Wie bildet man die reale Welt in eine akkurate und adäquate symbolische Beschreibung ab, unter Beachtung der zeitlichen Rahmenbedingungen für diese Abbildung?**
    Computersehen, Verstehen natürlicher Sprache, Maschinelles Lernen
- **Repräsentationsproblem:**
  - **Wie repräsentiert man symbolisch Information über komplexe Entitäten und Prozesse in der realen Welt?**
  - **Wie können Agenten unter Beachtung der zeitlichen Rahmenbedingungen über diese Information nützliche Schlussfolgerungen ziehen?**

  Wissensrepräsentation, Maschinelles Schließen und Planen
- **Keines dieser Probleme ist gelöst**
- **Tieferliegendes Problem:**
  - **Komplexität von Algorithmen zur Manipulation von Symbolen (insbesondere basierend auf Suchverfahren; oft nicht handhabbar)**

Deduktives Schließen

- **Idee: Entscheidung als Theorembeweisen**
- **Ausgangspunkt: Verwendung von Logik zur Kodierung der Umgebung sowie einer Theorie, die in jeder Situation die bestmögliche Aktion beschreibt.**
- **Sei**
  - **ρ eine Menge von Deduktionsregeln**
  - **Δ eine logische Datenbank, die den momentanen Zustand der Welt beschreibt (d.h., eine Menge logischer Formeln bzgl. einer Sprache L)**
  - **A die Menge der Aktionen, die der Agent ausführen kann**
- **Wir schreiben $\Delta \vdash_{\rho} \Phi$ dafür, dass Φ aus Δ bezüglich ρ folgt, d.h. mittels (ausschließlich Regeln aus) ρ bewiesen werden kann.**

SIEMENS

CORPORATE TECHNOLOGY

Aktionsselektion durch Theorembeweisen

- Beschreibung der Umgebung E (siehe Kapitel 2) durch eine Menge logischer Formeln (= Datenbasis Δ)
- Beschreibung der Aktionsselektionsfunktion *action* (s. Kapitel 2) durch eine Menge ρ von Deduktionsregeln
- Idee: Der Agentenprogrammierer beschreibt die Deduktionsregeln ρ und die Datenbasis Δ derart, dass gilt:

  WENN
  eine Formel Do(α) bewiesen werden kann, wobei α ein Term ist, der eine Aktion bedeutet
  DANN
  ist α die beste Aktion, die der Agent ausführen kann

SIEMENS

CORPORATE TECHNOLOGY

Einfacher Aktionsselektionsalgorithmus basierend auf Theorem-Beweisen

```
for each a ∈ Ac do
  if Δ ⊢ρ Do(a) then
      return a
  endif
endfor

for each a ∈ Ac do
  if Δ ⊬ρ ¬Do(a) then
      return a
  endif
endfor

return null
```

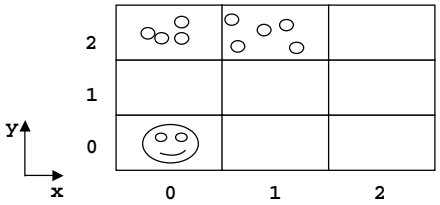
/* Versuche, eine explizit erlaubte Aktion zu finden */

/* Versuche, eine konsistente Aktion zu finden */

/* Keine Aktion gefunden, tue nichts */

Beispiel: Staubsauger-Roboter

- Umgebung des Roboters: Welt repräsentiert als zwei-dimensionales Gitternetz; Felder des Gitters enthalten Krümel
- Aufgabe des Roboters: Reinigen des gesamten Gitternetzes durch Aufsaugen der Krümel



- Symbolische Beschreibung der Welt durch Prädikate
  - *isAt(x, y)*: momentane Position des Roboters ist *(x, y)*
  - *dirtAt(x, y):* auf Position *(x, y)* gibt es Krümel
  - *facing(d):* der Roboter schaut in Richtung *d*
- Mögliche Aktionen:

$$Ac = \{turn\_right, forward, suck\}$$

SIEMENS

CORPORATE TECHNOLOGY

Aktualisierung des internen Zustandes

- Wir beschreiben die in Kapitel 2 definierte *next* Funktion, mithilfe derer der Agent seinen internen Zustand basierend auf seine aktuelle Wahrnehmung aktualisiert, für das Beispiel
- Dazu verwenden wir zwei Funktionen
  - *old* beschreibt die alte Information in der Datenbank, die durch die Funktion *next* gelöscht werden soll
    $old(\Delta)=\{W(t_1,...,t_n) \mid W \in \{isAt, dirtAt, facing\} \wedge P(t_1,...,t_n) \in \Delta\}$
  - *new* beschreibt die neue Information, die durch *next* zur Datenbank hinzugefügt werden soll, d.h., *new* erzeugt Instanzen der Prädikate *isAt(), dirtAt(), facing()*
- Dann ist *next* definiert als
  $next(\Delta, w) = (\Delta \setminus old(\Delta)) \cup new(\Delta, w)$

SIEMENS

CORPORATE TECHNOLOGY

Deduktionsregeln

- Entscheidungsregeln ρ:

  $\forall X,Y.\ isAt(X,Y) \wedge dirtAt(X,Y) \rightarrow Do(suck)$
  $isAt(0,0) \wedge facing(n) \wedge \neg\, dirtAt(0,0) \rightarrow Do(forward)$
  $isAt(0,1) \wedge facing(n) \wedge \neg\, dirtAt(0,1) \rightarrow Do(forward)$
  $isAt(0,2) \wedge facing(n) \wedge \neg\, dirtAt(0,2) \rightarrow Do(turn\_r)$
  $isAt(0,2) \wedge facing(e) \rightarrow Do(forward)$
  etc.
- Annahme: Der Roboter startet im Feld (0,0) des Gitternetzes und schaut nach Norden
- Mit den obengenannten fünf Regeln und der von uns definierten *next* Funktion wird der Roboter in der obigen Welt alle verschmutzten Felder reinigen und in Feld (1,2) anhalten.

SIEMENS

CORPORATE TECHNOLOGY

Agenten-orientierte Programmierung

- Grundlegendes Papier *Agent-oriented Programming* von Yoav Shoham (Artificial Intelligence, 1993)
- AOP: Neues Programmierparadigma, basierend auf einer Multiagentensicht
- Idee: Agenten werden direkt auf der Basis intentionaler Begriffe wie Wissen (*belief*), Zielen (*intentions*) und Verpflichtung (*commitment*) programmiert.
- Verwendung intentionaler Begriffe als nützliche Abstraktion zur Beschreibung komplexer Systeme, siehe Kapitel 1. („mein Auto will heute wieder nicht anspringen ... “).

SIEMENS

CORPORATE TECHNOLOGY

Komponenten eines AOP-Systems

- Nach Shoham besteht vollständiges AOP-System aus drei Komponenten
  - einer Logik zur Spezifikation von Agenten und ihrer mentalen Zustände
  - einer (interpretierten) Programmiersprache für Agenten
  - einem „Agentifizierungsmechanismus“, mit dem existierende Applikationen (z.B. Datenbanken) in Agenten verwandelt werden können.
- Das von Shoham beschriebene AOP-System AGENT0 realisiert nur die beiden ersten Komponenten, d.h. die Logik und Programmiersprache; diese sind verbunden durch eine Semantik

SIEMENS

CORPORATE TECHNOLOGY

AGENT0

Jeder Agent in AGENT0 hat vier Komponenten:

- **Eine Menge von Fähigkeiten (Aktionen)**
- **Eine Menge initialer Annahmen (Beliefs) des Agenten bezüglich seiner Umgebung**
- **Eine Menge initialer Commitments (Aktionen, auf deren Ausführung sich der Agent festgelegt hat)**
- **Eine Menge von Commitment-Regeln; diese Menge bestimmt, wie der Agent handelt.**

SIEMENS

CORPORATE TECHNOLOGY

Commitment-Regeln

- **Jede Commitment-Regel beinhaltet:**
  - **Eine Nachrichten-Bedingung (message condition)**
  - **Eine Zustandsbedingung (mental condition)**
  - **Eine Aktion**
- **Die Kontrollstruktur des Agenten ist beschrieben durch eine Iteration von sog. Agentenzyklen**
- **In jedem Zyklus wird für jede Commitment-Regel:**
  - **Die Nachrichtenbedingung mit den Nachrichten gematcht (verglichen), die der Agent erhalten hat**
  - **Die Zustandsbedingung mit den Annahmen des Agenten gematcht**
- **Wenn die Regel feuert (Nachrichtenbedingung und Zustandsbedingung sind erfüllt), erzeugt der Agent ein Commitment für die Aktion, die durch die Regel festgelegt wird.**

SIEMENS

CORPORATE TECHNOLOGY

Aktionstypen

- ***private*** **(intern ausgeführte Subroutine)**
- ***communicative*** **(versenden von Nachrichten)**

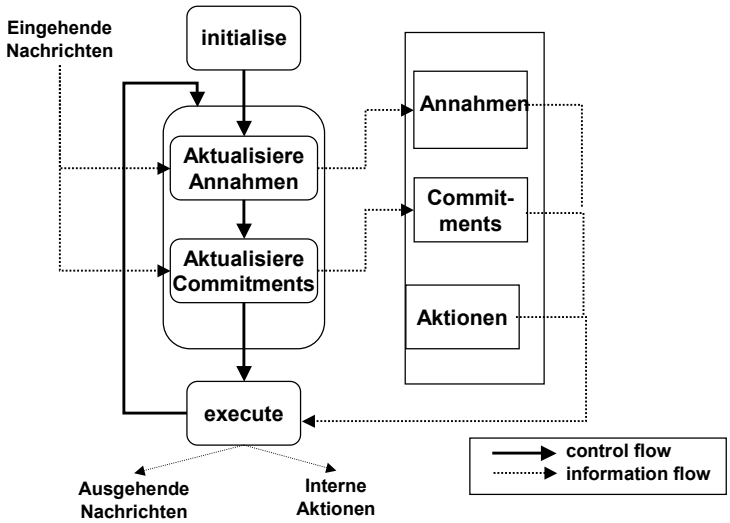
Nachrichtentypen

- ***requests*****, um einen Agenten zum Commitment zu einer Aktion zu bringen**
- ***unrequests*****, um von einem zuvor erhaltenen Commitment Abstand zu nehmen**
- ***informs*****, um Information weiterzugeben**

SIEMENS

CORPORATE TECHNOLOGY

Lebenszyklus eines Agenten in AGENT0



Beispiel: Commitmentregel in AGENT0

```
COMMIT(
  ( agent, REQUEST, DO(time, action) ), ;;; msg
  condition
  ( B,
    [     now, Friend agent] AND
          CAN(self, action) AND
     NOT [time, CMT(self, anyaction)]
  ), ;;; mental condition
  self,
  DO(time, action))
```


Und jetzt auf Deutsch ...

WENN ich eine Nachricht von *agent* erhalte mit der Anfrage, zur Zeit *time* eine Aktion *action* auszuführen, und ich glaube, dass:

- ***agent* momentan ein Freund ist UND**
- **Ich die Aktion *action* ausführen kann UND**
- **Ich mich für den Zeitpunkt *time* noch nicht auf eine andere Aktion festgelegt habe**

DANN lege ich mich fest, Aktion *action* zur Zeit *time* auszuführen

Bewertung von AGENT0

- AGENT0 unterstützt Kooperation und Kommunikation zwischen Agenten und bietet ein einfaches Debugging-Tool
- AGENT0 ist ein Prototyp, dessen Zweck der Nachweis der prinzipiellen Machbarkeit des AOP-Paradigmas war; AGENT0 ist keine kommerziell einsetzbare Programmiersprache
- Ein Manko von AGENT0 war das Fehlen von Planungsfähigkeiten
- PLACA (Planning Communicating Agents) [Thomas, 1995] ist eine Erweiterung von AGENT0 um diese Fähigkeit
- Die Programmierung von PLACA-Agenten erfolgt wie in AGENT0 durch Regeln, die den mentalen Zustand des Agenten modifizieren (sog. *mental change rules*)

Beispiel: Mental change rule in PLACA

```
(((self ?agent REQUEST (?t (xeroxed ?x)))
  (AND (CAN-ACHIEVE (?t (xeroxed ?x)))
       (NOT (BEL (*now* shelving)))
       (NOT (BEL (*now* (vip ?agent))))
  )
 )
 (ADOPT (INTEND (5pm (xeroxed ?x))))
 (?agent self INFORM
       (*now* (INTEND (5pm (xeroxed ?x))))
 )
)
```

SIEMENS

CORPORATE TECHNOLOGY

Und nochmal auf Deutsch ...

WENN jemand Dich fragt, ein Dokument für ihn zu kopieren, UND

Du bist dazu in der Lage UND

Du glaubst nicht, dass es sich bei dem Auftraggeber um eine wichtige Person handelt, ODER dass Du jetzt Bücher wegräumen solltest,

DANN

Lege Dich darauf fest, das Dokument um 5 Uhr nachmittags zu kopieren und informiere den Auftraggeber über dieses Ziel

SIEMENS

CORPORATE TECHNOLOGY

Probleme des deliberativen Ansatzes

- **Wie konvertiert man den Input einer Videokamera in ein Fakt „dirtAt(0,1)“?**
- **Der Entscheidungsprozess verlangt eine *statische* Umgebung, d.h. die Welt darf sich zwischen t und t+1 nicht ändern**
- **Verwendet man eine zur Beschreibung der Welt oder der Entscheidungsregeln prädikatenlogische Sprache erster Ordnung (d.h., mit existenz- und allquantifizierten Variablen), ist das Entscheidungsverfahren *unentscheidbar***
- **Selbst wenn Aussagenlogik verwendet wird, sind die Entscheidungsalgorithmen co-NP-vollständig im worst-case**
- **Im allgemeinen Fall, der Entscheidungen unter Begrenzung der Rechenzeit erfordert, ist damit der Ansatz nicht brauchbar**

Was kann man tun?

- Abschwächen der Logik (geringere Expressivität der logischen Sprache → geringere Komplexität der Entscheidungsverfahren)
- Verwendung nicht-logischer Wissensrepräsentationen
- (Teilweises) Durchführen der für die Entscheidungsfindung benötigten Berechnungen zur Designzeit des System, nicht zur Laufzeit
- Aktionsorientierte Arten des Schließens
- Im folgenden werden wir einige alternative Ansätze betrachten

3.2 Agenten für praktisches Schließen

- Rein logisches (d.h., wissensorientiertes Schließen) ist mit einer Reihe von Problemen verbunden, die es für praktische Ansätze nicht geeignet erscheinen lassen
- Alternativer Ansatz: „Praktisches Schließen“
- Praktisches Schließen verbindet die Fähigkeit zur Deliberation mit einem zielgerichteten Verhalten
- „Practical reasoning is a matter of weighing conflicting considerations for and against cometing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes“ [Bratman, 1990]

SIEMENS

CORPORATE TECHNOLOGY

Praktisches Schließen besteht aus zwei Aktivitäten

- ***Deliberation*** **(Was?)**

  Entscheidung, welche Ziele erreicht werden sollen
- ***Means-Ends Reasoning*** **(Wie?)**

  Entscheidung, auf welche Art und Weise diese Ziele erreicht werden sollen
- **Das Ergebnis des Deliberationsprozesses sind *Intentionen***

SIEMENS

CORPORATE TECHNOLOGY

Die Rolle von Intentionen im Praktischen Schließen

- **Intentionen stellen Aufgaben für Agenten da, auf deren Erfüllung (oder zumindest: auf den Versuch deren Erfüllung) sich die Agenten festlegen**
- **Bestehende Intentionen können die Annahme anderer Intentionen, die nicht mit ihnen konsistent (vereinbar) sind, blockieren.**

  Dies unterscheidet Intentionen von *Zielen* (engl. *Desire*)*:* ein Agent kann beliebig viele inkonsistente Ziele haben. Erst in dem Moment, wo er beschließt, ein Ziel aktiv zu verfolgen, nimmt er eine Intention an
- **Agenten verfolgen den Erfolg ihrer Intentionen und versuchen diese zu erfüllen.**

SIEMENS

CORPORATE TECHNOLOGY

Weitere Eigenschaften von Intentionen

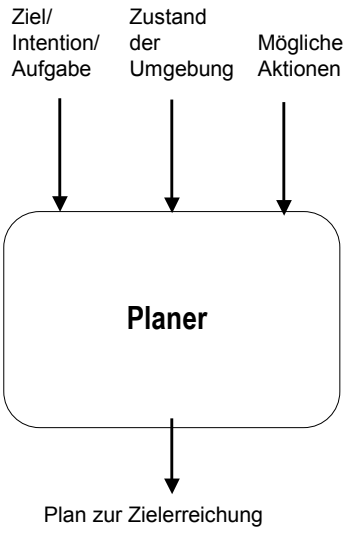
- Agenten glauben an die prinzipielle Möglichkeit der Erfüllung ihrer Intentionen
- Agenten glauben nicht, dass sie ihre Intentionen nicht erfüllen können (Rationalitätsannahme)
- Unter bestimmten Voraussetzungen glauben Agenten, dass sie ihre Intentionen erfüllen können
- Agenten müssen nicht alle Nebeneffekte ihrer Intentionen intendieren (z.B. die Schmerzen beim Zahnarzt...)

SIEMENS

CORPORATE TECHNOLOGY

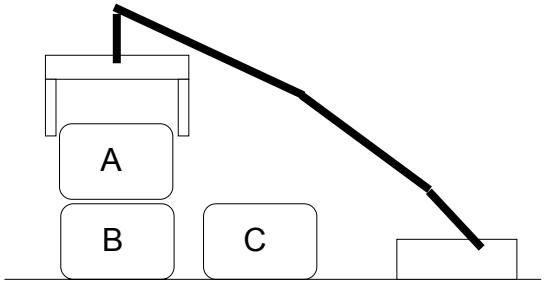
3.2.1 Means-Ends Reasoning

- Means-Ends = „von den Mitteln zum Ziel“
- Agent verfügt über:
  - Eine Repräsentation einer zu erfüllenden Aufgabe / Intention
  - Eine Repräsentation der Aktionen, die er ausführen kann
  - Eine Repräsentation seiner Umgebung
- Basierend auf diesem Wissen versucht er nun, einen Plan zu generieren, der die Aufgabe / Intention erfüllt
- Means-Ends Reasoning ist im wesentlichen identisch mit Automatische Programmierung oder Maschinellem Planen
- Maschinelle Planung ist ein gut erforschtes Zentralthema der Künstlichen Intelligenz seit den frühen Siebzigerjahren



- **Wie repräsentiert man**
  - **Zu erreichende Ziele?**
  - **Den Zustand der Umgebung?**
  - **Aktionen?**
  - **Pläne?**
- **Wir demonstrieren diese Techniken anhand eines Standard-Szenarios aus der KI: Der Blockswelt**

3.2.2 Planen in der Blockswelt



- **Beinhaltet einen Roboterarm, eine Anzahl von Blöcken (A, B, C, ...) gleicher Größe und eine Tischfläche**

SIEMENS

CORPORATE TECHNOLOGY

Repräsentation der Umgebung

- **Definition der Begriffswelt (Ontologie) durch Domänenprädikate**

On(x, y)	***Objekt x steht auf Objekt y***
OnTable(x)	***Objekt x steht auf der Tischplatte***
Clear(x)	***auf Objekt x steht kein anderes Objekt***
Holding(x)	***der Roboterarm hält x***
ArmEmpty	***der Roboterarm ist leer***

- **Repräsentation der Beispielumgebung:**

Clear(A)
Clear(C)
On(A, B)
OnTable(B)
OnTable(C)
ArmEmpty

SIEMENS

CORPORATE TECHNOLOGY

- **Anwendung der *Closed World Assumption:* wir nehmen an, dass alles, was nicht explizit beschrieben ist, falsch ist, d.h., nicht gilt**
- **Wir definieren nun ein *Ziel* als eine Menge von Formeln, z.B.**

OnTable(A), OnTable(B), OnTable(C)

SIEMENS

CORPORATE TECHNOLOGY

Aktionen in STRIPS

- Aktionen werden mithilfe ein erstmals im STRIPS Planungs-system (Fikes und Nilsson) eingesetzten Methode repräsentiert
- Jede Aktion ist beschrieben durch
  - Einen Namen
  - Argumente
  - Eine PRE-Liste: Liste mit Fakten, die gelten müssen, damit die Aktion ausgeführt werden kann (Vorbedingungen)
  - Eine DELETE-Liste: Liste mit Fakten, die nach Ausführung der Aktion nicht länger gültig sind
  - Eine ADD-Liste: Liste mit Fakten, die durch die Ausführug der Aktion nwahr werden
  - Alle Fakten können Variablen enthalten
- Die DELETE- und ADD-Liste beschreiben die Effekte der Aktion

SIEMENS

CORPORATE TECHNOLOGY

Beispielaktionen

- Abstellen eines im Roboterarm befindlichen Blocks auf einen anderen Block

  Stack(x, y)
  PRE *{Clear(y), Holding(x)}*
  DELETE *{Clear(y), Holding(x)}*
  ADD *{ArmEmpty, On(x, y)}*

Aufnehmen eines Blocks vom Tisch

  Pickup(x)
  PRE *{ArmEmpty, OnTable(x), Clear(x)}*
  DELETE *{ArmEmpty, OnTable(x)}*
  ADD *{Holding(x)}*

Definitionen

- Eine Aktion $a \in Ac$ ist ein Tripel $a = (P_a, D_a, A_a)$, mit
  - P_a, D_a, A_a Mengen logischer Formeln erster Ordnung
  - P_a beschreibt Vorbedingungen von a (PRE-Liste)
  - D_a beschreibt die Fakten, die durch die Ausführung von a falsch werden (DELETE-Liste)
  - A_a beschreibt die Fakten, die durch die Ausführung von a wahr werden (ADD-Liste)
- Planungsproblem (über Ac) ist ein Tripel (Δ, O, γ), *mit*
  - Δ sind die Beliefs des Agenten über den initialen Zustand der Umgebung (= Menge von Fakten)
  - $O = \{(P_a, D_a, A_a) \mid a \in Ac\}$ ist eine indexierte Menge von Operatorbeschreibungen, eine für jede Aktion in Ac
  - γ ist eine Menge logischer Formeln erster Ordnung, die das Ziel/die Aufgabe/Intention beschreibt, die erfüllt werden soll

Definition Plan

- Ein (linearer) Plan π ist dann eine Folge von Aktionen $\pi = (a_1, \ldots, a_n)$ mit $a_i \in Ac$, wobei alle Variablen durch Konstanten ersetzt sind
- Bezüglich eines Planungsproblems (Δ, O, γ) induziert ein Plan eine Folge von $n+1$ Umgebungsmodellen $\Delta_0, \Delta_1, \ldots, \Delta_n$, wobei gilt:

$$\Delta_0 = \Delta$$

und

$$\Delta_i = (\Delta_{i-1} / D_{a_i}) \cup A_{a_i} \text{ für } 1 \leq i \leq n$$

SIEMENS

CORPORATE TECHNOLOGY

Anwendbarkeit und Korrektheit von Plänen

- Ein Plan π heißt anwendbar auf ein Planungsproblem (Δ, O, γ) gdw. die Vorbedingung jeder Aktion durch das jeweilige Umgebungsmodell erfüllt ist, d.h., wenn

  $$\Delta_{i-1} \models P_{a_i} \text{ für } 1 \leq i \leq n$$

- Ein Plan π heißt korrekt bzgl. (Δ, O, γ) gdw. Er
  (1) anwendbar ist und
  (2) $\Delta_n = \gamma$, d.h., das Ziel im durch den Plan erreichten Endzustand erreicht ist
- Aufgabe eines Planers: gegeben ein Planungsproblem (Δ, O, γ), finde einen korrekten Plan für (Δ, O, γ) oder gib zurück, dass keiner existiert.

SIEMENS

CORPORATE TECHNOLOGY

Weitere Definitionen

- Die Vorbedingung eines Plans π bezeichnen wir mit $pre(\pi)$
- Für einen linearen Plan π ist $empty(\pi) = true$ gdw. π ist die leere Folge von Aktionen
- *execute(...)* ist eine Prozedur, die einen Plan als Eingabe hat und jede Aktion in dem Plan nacheinander ausführt
- Für einen Plan $\pi = (a_1, ..., a_n)$ ist hd(π) der Plan, der aus der ersten Aktion von $\pi = (a_1, ..., a_n)$ besteht: $hd(\pi) = (a_1)$; analog ist $tail(\pi)$ der Plan ohne die erste Aktion: $tail(\pi) = (a_2, ..., a_n)$
- Im folgenden bezeichnen $I \subseteq Int$ eine Menge von Intentionen, $B \subseteq Bel$ eine Menge von Fakten, die der Agent glaubt (= Beliefs), und $D \subseteq Des$ eine Menge von Optionen (= Desire), die der Agent hat
- Die Funktion plan: $\wp(Bel) \times \wp(Int) \times \wp(Ac) \rightarrow Plan$ repräsentiert das Means-Ends Reasoning des Agenten
- Plankorrektheit bzgl. Intentionen und Beliefs: $sound(\pi, I, B)$

3.2.3 Implementierung eines Agenten für Praktisches Schließen

- **Ziel: Beschreibung des Kontrollalgorithmus für einen solchen Agenten (Kontrollschleife)**

```
Kontrollalgorithmus PS_1

1.while true do
2.     beobachte die Umgebung
3.     aktualisiere internes Umgebungsmodell
4.     wähle die als nächstes zu erfüllende
       Intention aus (Deliberation)
5.     konstruiere mittels Means-Ends Reasoning
       einen Plan für die Intention
6.     führe den Plan aus
7.end-while
```

Wir konzentrieren uns in der Folge auf die Schritte 4 und 5

Kalkulative Rationalität

- **Problem: Deliberation und Means-Ends-Reasoning verbrauchen Berechnungsressourcen (Zeit!)**
- **Beispiel:**
  t0: Agent beginnt Deliberation
  t1: Agent beginnt Means-Ends-Reasoning
  t2: Agent beginnt Planausführung
- **Unter der Annahme, dass die Deliberationsfunktion des Agenten optimal ist (d.h., bei der Intentionsselektion den erwarteten Nutzen optimiert), hat der Agent zum Zeitpunkt t1 eine Intention ausgewählt, deren Erreichung zum Zeitpunkt t0 optimal gewesen wäre**
- **Dieses Verhalten nennt man kalkulative Rationalität**

SIEMENS

CORPORATE TECHNOLOGY

Wann sind kalkulativ rationale Agenten optimal?

- Wenn Deliberation und Means-Ends-Reasoning eine vernachlässigbare Zeit benötigen; ODER
- Die Umgebung während dieser Prozesse sich nicht verändert; ODER
- Wenn eine Intention, die zu dem Zeitpunkt optimal ist, zu dem der Agent die Welt beobachtet hat, dies auch noch zu dem Zeitpunkt ist, wo der Agent mit der Ausführung der gewählten Aktion beginnt.

SIEMENS

CORPORATE TECHNOLOGY

3.2.4 Deliberation

- Im folgenden werden wir uns den Deliberationsmechanismus unseres Agenten für praktisches Schließen näher betrachten und einige damit zusammenhängende Probleme erörtern
- Deliberationsprozess
  - Berechne Optionen für mögliche Intentionen (Funktion *options()*)
  - Treffe Auswahl zwischen diesen Optionen (Funktion *filter()*)
  - Erzeuge Commitment (Festlegung) auf eine oder mehrere daraus
- Intentionen sind also Optionen, die ein Agent gewählt hat

SIEMENS

CORPORATE TECHNOLOGY

Verfeinerter Kontrollalgorithmus PS_2

```
1. B := B0;
2. I := I0;
3. while true do
4.     w := nächstes wahrgenommenes Fakt;
5.     B := brf(B, w); // belief revision function
6.     D := options(B, I);
7.     I := filter(B, D, I);
8.     π := plan(B, I, Ac);
9.     execute(π);
10.end-while
```


SIEMENS

CORPORATE TECHNOLOGY

Festlegungsstrategien für Intentionen

- **Fragestellung: Wie lange verfolgt ein Agent eine Intention?**
- **Anmerkung: Es gibt Intentionen, die sich auf das WAS beziehen (d.h., Zielzustände), und Intentionen, die sich auf das WIE beziehen (die Mittel, mit denen der Zielzustand erreicht werden soll)**
- **Ein Agent nach Kontrollalgorithmus PS_2 verfolgt eine „sture" Festlegungsstrategie, sowohl bezüglich seiner Ziele als auch bezüglich seiner Mittel**
- **Im folgenden beschreiben wir eine flexiblere Variante: der Agent plant von neuem, wenn ein Plan fehlschlägt. Hierzu erweitern wir die *execute()* Prozedur des Agenten**

SIEMENS

CORPORATE TECHNOLOGY

Kontrollalgorithmus PS_3

```
1. B := B0;
2. I := I0;
3. while true do
4.         w := nächstes wahrgenommenes Fakt;
5.         B := brf(B, w); // belief revision fct.
6.         D := options(B, I);
7.         I := filter(B, D, I);
8.         π := plan(B, I, Ac);
9.         while not empty(π) do
10.              a := hd(π);
11.              execute(a);
12.              π := tail(π);
13.              w := nächstes wahrgenommenes Fakt;
14.              B := brf(B, w);
15.              if not sound(π, I, B)
16.                π := plan(B, I, Ac);
17.              end-if
18.        end-while
19. end-while
```


SIEMENS

CORPORATE TECHNOLOGY

- Mit PS_3 kann der Agent die Wahl seiner Mittel flexibler an Änderungen seiner Umwelt anpassen
- Aber: Er hält unentwegt an einmal ausgewählten Intentionen fest
- Algorithmus PS_3 führt immer noch zu unflexiblem Verhalten des Agenten
- Wir modifizieren den Kontrollalgorithmus um eine einfache Strategie zur periodischen Re-Evaluation von Intentionen
- In jeder Iteration prüft der Agent, ob die Intention erfüllt ist, oder ihre Erfüllung unmöglich ist.

SIEMENS

CORPORATE TECHNOLOGY

Kontrollalgorithmus PS_4

```
1. B := B0;
2. I := I0;
3. while true do
4.         w := nächstes wahrgenommenes Fakt;
5.         B := brf(B, w); // belief revision fct.
6.         D := options(B, I);
7.         I := filter(B, D, I);
8.         π := plan(B, I, Ac);
9.         while not (empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) do
10.                a := hd(π);
11.                execute(a);
12.                π := tail(π);
13.                w := nächstes wahrgenommenes Fakt;
14.                B := brf(B, w);
15.                if not sound(π, I, B)
16.                  π := plan(B, I, Ac);
17.                end-if
18.        end-while
19. end-while
```


SIEMENS

CORPORATE TECHNOLOGY

Überprüfung von Intentionen

- **Mit Strategie PS_4 prüft ein Agent seine Intentionen wenn**
  - **Er einen Plan zur Erreichung seiner momentanen Intention vollständig abgearbeitet hat; oder**
  - **Er glaubt, dass er seine Intentionen erreicht hat; oder**
  - **Er glaubt, dass seine Intentionen nicht erreichbar sind**
- **Wir erweitern den Algorithmus, so dass die Intentionen nach der Ausführung jeder Aktion überprüft werden**

Kontrollalgorithmus PS_5

```
1. B := B0;
2. I := I0;
3. while true do
4.         w := nächstes wahrgenommenes Fakt;
5.         B := brf(B, w); // belief revision fct.
6.         D := options(B, I);
7.         I := filter(B, D, I);
8.         π := plan(B, I, Ac);
9.         while not empty(π)
                  or succeeded(I, B)
                  or impossible(I, B) do
10.                a := hd(π);
11.                execute(a);
12.                π := tail(π);
13.                w := nächstes wahrgenommenes Fakt;
14.                B := brf(B, w);
15.                D := options(B, I);
16.                I := filter(B, D, I);
17.                if not sound(π, I, B)
18.                  π := plan(B, I, Ac);
19.                end-if
20.        end-while
21. end-while
```


- **Überprüfung von Intentionen verursacht Kosten**
- **Zu seltenes Überprüfen kann dazu führen, dass ein Agent nicht rechtzeitig bemerkt, dass ein Ziel erfüllt oder nicht länger erreichbar ist**
- **Zu häufiges Überprüfen kann dazu führen, dass der Agent nicht genug Zeit auf die eigentliche Erfüllung seiner Ziele verwendet**
- **Idee: Metalevel-Kontroll-Algorithmus** *reconsider()***, der entscheidet, wann und wie oft Intentionen überprüft werden**
- **Wichtige Annahme:**
  Kosten für Überprüfung << Kosten des Deliberationsprozesses

SIEMENS

CORPORATE TECHNOLOGY

Kontrollalgorithmus PS_6

```
1. B := B0;
2. I := I0;
3. while true do
4.         w := nächstes wahrgenommenes Fakt;
5.         B := brf(B, w); // belief revision fct.
6.         D := options(B, I);
7.         I := filter(B, D, I);
8.         π := plan(B, I, Ac);
9.         while not empty(π)
                  or succeeded(I, B)
                  or impossible(I, B) do
10.              a := hd(π);
11.              execute(a);
12.              π := tail(π);
13.              w := nächstes wahrgenommenes Fakt;
14.              B := brf(B, w);
15.              if reconsider(I, B)
16.                D := options(B, I);
17.                I := filter(B, D, I);
18.              end-if
19.              if not sound(π, I, B)
20.                π := plan(B, I, Ac);
21.              end-if
22.        end-while
23. end-while
```


SIEMENS

CORPORATE TECHNOLOGY

Beschränkungen des Kontrollalgorithmus

- **Der Kontrollalgorithmus, den wir bis hierhin entwickelt haben, weist noch einige Einschränkungen auf und muss u.U. für praktische Anwendungen angepasst werden, z.B. bezüglich:**
- **Nebenläufige Perzeption und Aktion**
- **Abhängigkeiten zwischen einzelnen Zielen / Intentionen**
- **Aktive Sensorik: Sensorik als Aktion**
- **Kontinuierliche Berechnungsmodelle**
- **Nichtlineare Pläne**
- **Berechenbarkeit der Funktionen *brf(), options(), filter(), plan()*???**

SIEMENS

CORPORATE TECHNOLOGY

3.2.5 Belief, Desire, Intention Semantik

- Fragestellung: wie kann man Agenten, die auf der Basis von Belief, Desire, Intention-Architekturen beschrieben sind, formal beschreiben (→ Semantik)
- Rao & Georgeff (1991): BDI-Logik als nicht-klassische Logik mit modalen Konnektiven für Belief, Desire und Intentionen
- Basislogik ist eine Erweiterung der Logik CTL*; CTL* basiert auf einem Zeitmodell mit Verzweigungen, d.h., es können mehrere mögliche Welten in der Zukunft modelliert werden.

SIEMENS

CORPORATE TECHNOLOGY

Syntax der BDI Logik

- Aus der klassischen Logik: $\neg, \wedge, \vee$, ...
- CTL* Quantoren über Pfade
  - Aϕ „auf allen Pfaden gilt ϕ“
  - Eϕ „es existiert ein Pfad, auf dem ϕ gilt“
- BDI Konnektiven
  - (Bel i ϕ) i glaubt ϕ
  - (Des i ϕ) i hat ϕ als Ziel
  - (Int i ϕ) i intendiert ϕ

Semantik der BDI-Logik

- Grundidee: „Mögliche Weltensemantik“ (Hintikka, 1962) (Kripke, 1969)
- Semantik von BDI-Konstrukten ist durch eine sogenannte Erreichbarkeitsrelation zwischen „Welten“ beschrieben. Eine Welt beschreibt einen möglichen Zustand. Durch unvollständige Information sind zu einem Zeitpunkt für den Agenten mehrere Zustände möglich
- Eine genauere Beschreibung der BDI-Logik und ihrer Semantik übersteigt den Rahmen dieser Vorlesung. Einen guten Überblick bietet (Wooldridge, 2002, Kapitel 12)
- Im folgenden betrachten wir noch einige mögliche Axiome der BDI-Logik und untersuchen, inwieweit BDI-Architekturen diese erfüllen können

Mögliche Eigenschaften von BDI-Architekturen

- Sei im folgenden
  - α eine O-Formel, d.h., eine Formel, die keine positiven Vorkommen des A-Quantors enthält
  - ϕ eine beliebige Formel
- Kompatibilität von Beliefs und Zielen
  $(\text{Des } \alpha) \Rightarrow (\text{Bel } \alpha)$
  D.h. alle Ziele (=Optionen) hält der Agent prinzipiell für möglich (s. *options()* Funktion)
- Kompatibilität von Zielen und Intentionen
  $(\text{Int } \alpha) \Rightarrow (\text{Des } \alpha)$
  D.h., alle Intentionen sind auch Ziele (s. *deliberate()* Funktion)
- Bewußtsein bzgl. Aktionen
  $\text{done}(\alpha) \Rightarrow \text{Bel}(\text{done}(\alpha))$

Weitere mögliche Eigenschaften

- Unverzügliche Ausführung von Intentionen
  $(\text{Int does}(\alpha)) \Rightarrow \text{does}(\alpha)$
  Wenn der Agent die Ausführung einer Intention im nächsten Zyklus beschlossen hat, wird er sie auch ausführen
  s. execute() Funktion
- Bewußtsein bzgl. Zielen und Intentionen
  $(\text{Des } \phi) \Rightarrow (\text{Bel } (\text{Des } \phi))$
  $(\text{Int } \phi) \Rightarrow (\text{Bel } (\text{Int } \phi))$
- Keine unendliche Verzögerung der Ausführung
  $(\text{Int } \phi) \Rightarrow A \Diamond(\neg(\text{Int } \phi))$
  Agent wird bzgl. einer Intention irgendwann handeln oder sie aufgeben

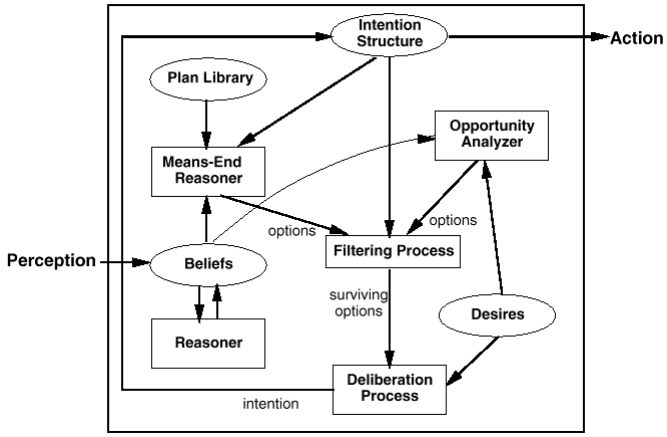
3.2.6 Implementierte BDI Agenten: IRMA

- Das Wissen eines Agenten im IRMA System (Bratman et al., 1987) ist durch vier Datenstrukturen beschrieben:
  - Eine Planbibliothek
  - Explizite Repräsentationen von Beliefs, Optionen und Intentionen
- Die Kontrollstruktur des Agenten ist gegeben durch
  - Eine Inferenz-Engine, um Schlüsse über die Umgebung zu ziehen
  - Einen Mean-Ends-Planer, um Pläne zu bestimmen, mit denen Intentionen erreicht werden können
  - Einen Options-Analysierer, die die Umgebung beobachtet und als Ergebnis von Änderungen neue Optionen erzeugt
  - Einen Filterprozess, der die Optionen ermittelt, die kompatibel mit den momentanen Intentionen des Agenten sind
  - Einen Deliberationsprozess, der die "besten" Intentionen auswählt

SIEMENS

CORPORATE TECHNOLOGY

IRMA Agentenarchitektur (Bratman et al, 1987)



SIEMENS

CORPORATE TECHNOLOGY

3.2.7: Das Procedural Reasoning System (PRS)

- PRS (Georgeff&Lansky, 1987) ist die vielleicht ausgereifteste Implementierung einer BDI-Architektur mit zahlreichen Anwendungen und Weiterentwicklungen (z.B., dMARS (d'Inverno et al, 1997), Jack (Busetta et al, 2000), JAM (Huber, 1999))
- Jeder PRS-Agent verfügt über eine Planbibliothek, die das prozedurale Wissen des Agent repräsentiert. Diese beschreibt das Wissen über Mechanismen, die der Agent zur Erfüllung seiner Intentionen anwenden kann
- Intentionen werden durch den „Laufzeitstack" des PRS-Interpreters repräsentiert
- Beliefs, Ziele und Intentionen sind explizit repräsentiert

SIEMENS

CORPORATE TECHNOLOGY

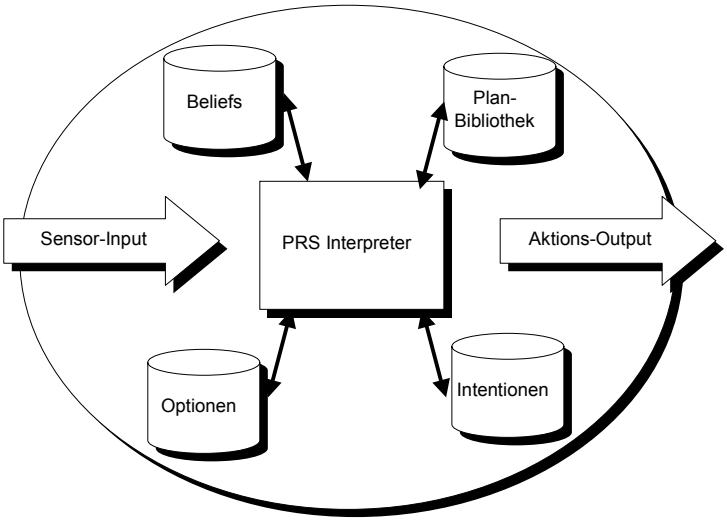
PRS

- **Ein Plan besteht aus:**
  - **einem Ziel (Nachbedingung des Plans)**
  - **einem Kontext (Vorbedingung) und**
  - **einem Körper, der die auszuführenden Aktionen beschreibt**
- **Zwei implementierte Alternativen zur Deliberation**
  - **Meta-Level-Pläne, d.h., ein Plan beschreibt, welcher Plan auszuwählen ist**
  - **Zuordnung von Nutzenwerten zu Plänen: Agent wählt den Plan mit dem höchsten Nutzen**

SIEMENS

CORPORATE TECHNOLOGY

PRS Agentenarchitektur



Beispiel: Modellierung eines Blockswelt-Szenarios in JAM

```
GOALS:
    ACHIEVE blocks_stacked;
FACTS:
    FACT ON „Block5“  „Block4“;
    FACT ON „Block4“ „Block3“;
    FACT ON „Block1“ „Block2“;
    FACT ON „Block2“ „Table“;
    FACT ON „Block3“ „Table“;
    FACT CLEAR „Block1“;
    FACT CLEAR „Block5“;
    FACT CLEAR „Table“;
PLAN: {
    NAME: „Top-Level Plan“;
    GOAL: ACHIEVE blocks_stacked;
    BODY:  ACHIEVE ON „Block3“ „Table“;
           ACHIEVE ON „Block2“ „Block3“;
           ACHIEVE ON „Block1“ „Block2“;
}
```

```
PLAN: {
    NAME: „Stack blocks“;
    GOAL: ACHIEVE ON $O1 $O2;
    BODY:  ACHIEVE CLEAR $O1;
           ACHIEVE CLEAR $O2;
           PERFORM move $O1 $O2;
    UTILITY: 10;
FAILURE: EXECUTE print „Plan failed!“;
}

PLAN: {
    NAME: „Clear a block“;
    GOAL: ACHIEVE CLEAR $O;
    CONTEXT: FACT ON $O2 $O;
    BODY:  ACHIEVE CLEAR $O2;
           ACHIEVE ON $O2 „Table“;
    EFFECTS: RETRACT ON $O2 $O;
    FAILURE: EXECUTE print „Plan failed!“;
}
```


SIEMENS

CORPORATE TECHNOLOGY

3.3 Architekturen für reaktive Agenten

- Mit der symbolischen KI sind zahlreiche ungelöste und (wahrscheinlich) auch unlösbare Probleme verbunden
- Seit den 1980-ern: Suche nach alternativen Ansätzen, inspiriert von Verhaltenspsychologie und Biologie
- Einfache interne Repräsentation der Welt
- Enge Verzahnung von Perzeption und Aktion
- Einfache Entscheidungsregeln
- Simon (1981): Die Komplexität des Verhaltens eines Agenten kann auch eine Reflektion der Komplexität der Umgebung des Agenten sein, nicht seines komplexen internen Designs
- Modelle: Ameisenkolonie, Bienenschwarm (Schwarmintelligenz)

SIEMENS

CORPORATE TECHNOLOGY

Rodney Brooks‘ Thesen

- Intelligentes Verhalten kann ohne explizite Repräsentation erzeugt werden
- Intelligentes Verhalten kann ohne explizites abstraktes Schließen erzeugt werden
- Intelligentes Verhalten ist eine emergente Eigenschaft gewisser komplexer Systeme
- Zwei Schlüsselideen:
  - *Situiertheit*: ‚Reale‘ Intelligenz ist situiert in der Welt, nicht körperlos wie in Theorembeweisern
  - *Emergenz*: Intelligenz entsteht durch Interaktion mit der Umgebung, ist also eine externe, keine interne isolierte Eigenschaft

SIEMENS

CORPORATE TECHNOLOGY

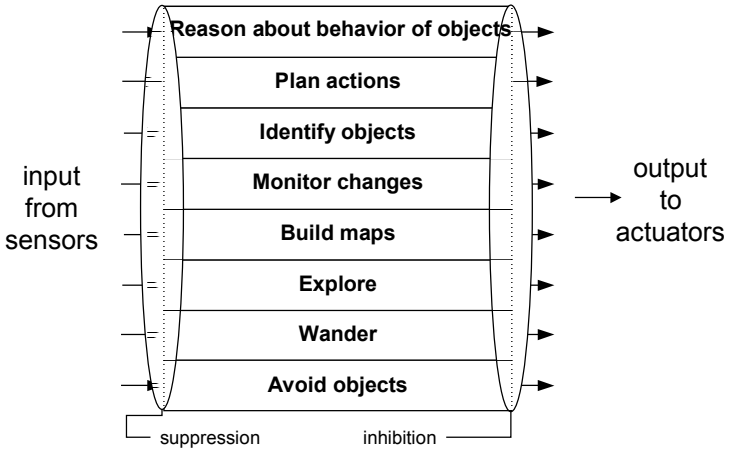
Subsumptionsarchitektur

- Um seine Ideen zu illustrieren, bauten Brooks und sein Team einige Agenten (Roboter) basierend auf der *Subsumptions-architektur*
- Hierarchie aufgabenorientierter Verhaltensmodule
- Jedes Verhaltensmodul ist eine einfache, regelartige Struktur
- Jedes Verhaltensmodul konkurriert mit anderen darum, die Kontrolle über den Agenten auszuüben
- Untere Ebenen repräsentieren primitivere Verhalten und haben Vorrang über die Verhaltensmodule auf höheren Ebenen
- Wenig Berechnungsaufwand, recht respektable Ergebnisse

SIEMENS

CORPORATE TECHNOLOGY

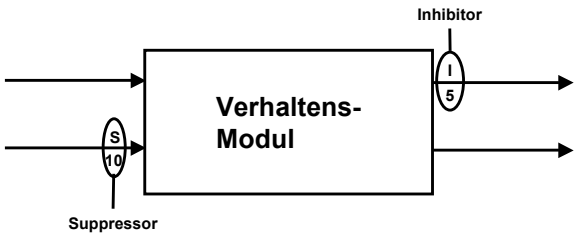
Subsumptionsarchitektur (Brooks, 1991)



SIEMENS

CORPORATE TECHNOLOGY

Kontrollmechanismen: Suppression und Inhibition



- **S: Signal unterdrückt und ersetzt die Eingabe in das Modul für einen Zeitraum**
- **I: Signal verhindert Ausgabe des Moduls für einen Zeitraum**
- **Suppressions- und Inhibitionsbeziehungen zwischen Modulen sind festverdrahtet**

SIEMENS

CORPORATE TECHNOLOGY

Situierte Automaten

- Rosenschein und Kaelbling (1986)
- Grundidee: Unterschiedliche Modellierungsparadigmen auf Konzeptions- und Implementierungsebene
- Agent wird mit einer deklarativen (regelbasierten) Sprache beschrieben; diese Spezifikation wird dann in eine digitale Maschine kompiliert, die die deklarative Spezifikation erfüllt
- Diese digitale Maschine operiert dann in einer beweisbaren zeitlichen Schranke
- Der Berechnungsaufwand des Schließens wird offline, zur Compile-Zeit statt zur Laufzeit durchgeführt

SIEMENS

CORPORATE TECHNOLOGY

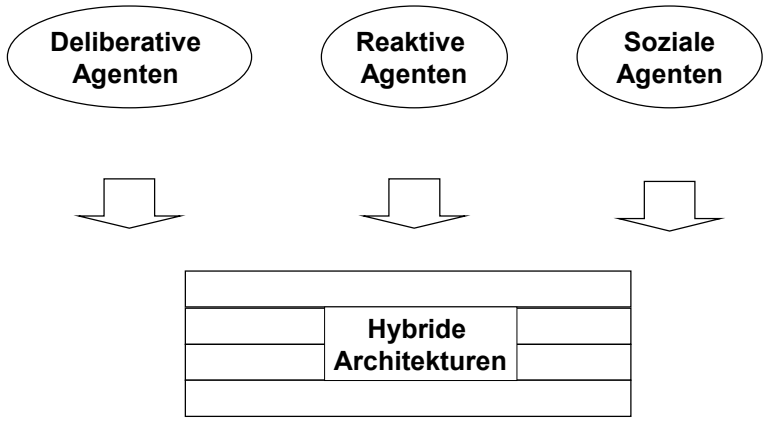
Bewertung Situierte Automaten

- Der Ansatz wirkt auf den ersten Blick sehr attraktiv, weil er die Stärken reaktiver und symbolisch-deklarativer Ansätze kombiniert
- Aber ...
  - Theoretischen Grenzen des Ansatzes sind noch unklar
  - Das Kompilieren einer Agentenspezifikation ist äquivalent zu einem NP-vollständigen Problem (für aussagenlogische Spezifikations-sprache)
  - Je ausdrucksstärker die Agentenspezifikationssprache, desto schwie-riger ist die Kompilation (ab einer gewissen Ausdrucksmächtigkeit ist sie unmöglich)

SIEMENS

CORPORATE TECHNOLOGY

3.4 Hybride Architekturen



SIEMENS

CORPORATE TECHNOLOGY

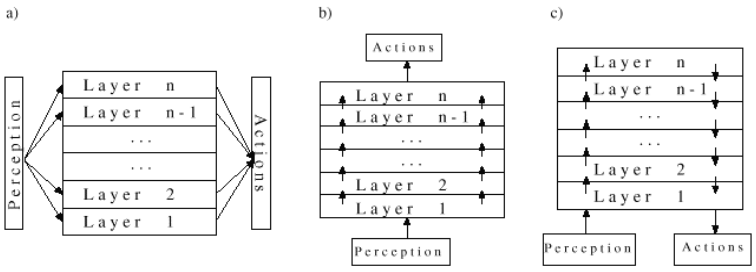
Hybride Architekturen

- Viele Forscher vertreten die These, dass weder ein vollständig deliberativer noch ein vollständig reaktiver Ansatz geeignet ist, um intelligente Agenten zu bauen
- Idee: Hybride Systeme, die die Vorteile beider Welten vereinen
- Häufig verwendeter Ansatz: Agent wird aus mehreren Komponenten aufgebaut
- Die Komponenten sind durch eine Kontrollstruktur verbunden
- Häufig: *Mehrschichtenarchitekturen*

Typischer Aufbau von Mehrschichtenarchitekturen

- Reaktive Ebene reagiert auf Ereignisse ohne komplexes Schließen
- Deliberative Ebene beinhaltet ein symbolisches Weltmodell, das Pläne entwickelt und Entscheidungen trifft
- Soziale oder Meta-Modell-Ebene, die Modelle von anderen Agenten enthält, Meta-Reasoning über die eigene Deliberation betreibt, oder Kooperation und Koordination mit anderen Agenten unterstützt
- Schlüsselprobleme in Mehrschichtenarchitekturen:
  - Wie sieht die Kontrollstruktur aus, die die Interaktionen der Ebenen steuert?
  - Wie kann ein kohärentes Gesamtverhalten des Agenten erreicht werden?

Ausprägungen von Kontrollstrukturen



- Horizontale Schichtung (a)
  - Alle Ebenen haben Zugriff zu Aktion und Wahrnehmung
  - Beispiel: Touring Machines
- Vertikale Schichtung: (b), (c)
  - Zugang zu Wahrnehmung und Aktion beschränkt
  - Beispiele: MECCA (b), InteRRaP (c)

SIEMENS

CORPORATE TECHNOLOGY

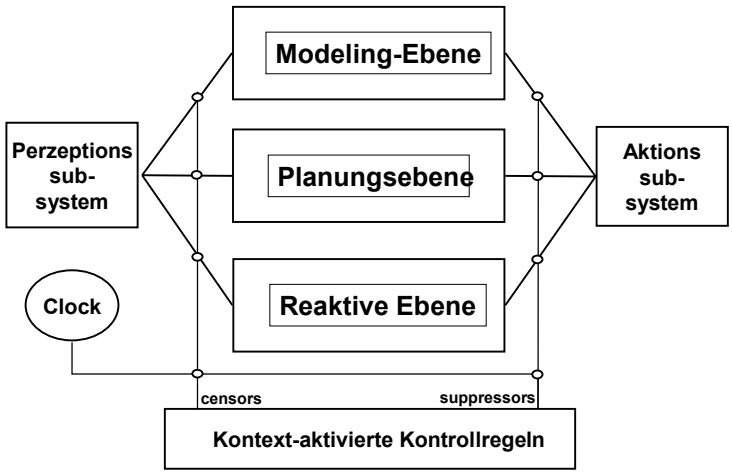
TouringMachines Architektur

- Ferguson, 1992
- Separate Subsysteme für Perzeption und Aktion, die direkt mit der Umgebung des Agenten in Verbindung stehen
- Drei Kontrollebenen
- Ein Kontrollframework, das die drei Kontrollebenen koordiniert

SIEMENS

CORPORATE TECHNOLOGY

TouringMachines (Ferguson, 1992)



SIEMENS

CORPORATE TECHNOLOGY

TouringMachines Kontrollebenen

- Reaktive Ebene: realisiert durch Menge von Situations-Aktions-Regeln, ähnlich wie bei der Subsumptionsarchitektur. Beispiel:

```
rule-1: obstacle-avoidance
     if
        is-in-front(Obstacle, Observer) and
        speed(Observer) > 0 and
        separation(Obstacle, Observer) < 5
     then
        changeOrientation(45)
```

- Planungsebene: repräsentiert die Ziele des Agenten;konstruiert Pläne (mit Hilfe einer Planbibliothek) und wählt Aktionen aus, um die Ziele zu erfüllen
- Modeling-Ebene: beinhaltet Informationen über den „kognitiven Zustand“ anderer Entitäten in der Umgebung des Agenten

SIEMENS

CORPORATE TECHNOLOGY

TouringMachines Kontrollstruktur

- TouringMachines-Ebenen sind getaktet, d.h., zu jedem Takt kann jede Ebene die Wahrnehmung aktualisieren und die Ausführung einer Aktion anstoßen
- Die drei Ebenen kommunizieren mit einander und sind in eine Kontrollstruktur eingebettet, die Eingabe und Ausgabe der Ebenen filtern kann und dabei zweierlei Arten von Kontrollregeln verwendet
- Zensor-Regeln: Unterdrücken die Wahrnehmung einer Ebene
- Suppressor-Regeln: Unterdrücken Aktion einer Komponente

SIEMENS

CORPORATE TECHNOLOGY

Beispiel: Zensoren und Suppressoren

Zensor:

```
censor-rule-1:
  IF entity(obstacle-6) ∈ PerceptionBuffer
  THEN
   remove-sensory-record(reactive-layer, entity(obstacle-
  6))
```

Suppressor:

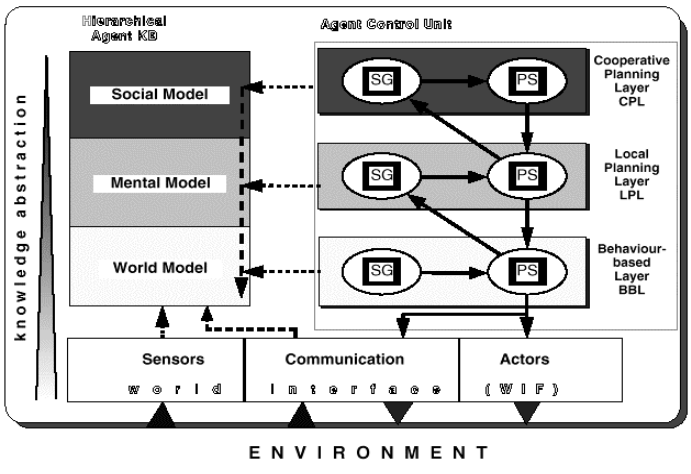
```
suppressor-rule-1:
  IF action-cmd(RL-rule-6, change-orientation)
   in ActionBuffer
  AND current-intention(start-overtake)
  THEN
   remove-action-cmd(reactive-layer, change-orientation)
```

Im zweiten Beispiel ist `RL-rule-6` ein Verhalten, den Roboter daran zu hindern, die Wegmarkierungslinien zu überqueren

SIEMENS

CORPORATE TECHNOLOGY

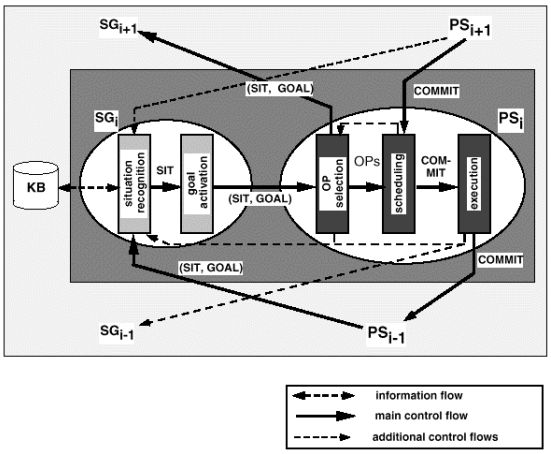
InteRRaP (Müller, 1996)



SIEMENS

CORPORATE TECHNOLOGY

Struktur einer InteRRaP Kontrollebene



SIEMENS

CORPORATE TECHNOLOGY

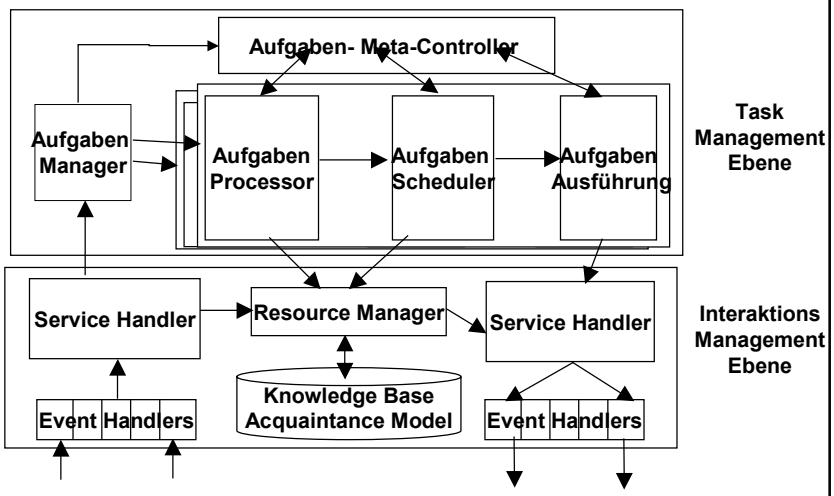
Architekturen für Internetagenten

- Neue Anforderungen an Softwareagenten im Internet
- Serviceorientierung: Agenten bieten Dienste (Services) an
- Nebenläufigkeit: Agenten bieten mehrere Dienste und mehrere Instanzen eines Dienstes zur gleichen Zeit an
- Andere Aspekte wie Flexibilität und Mobilität sind ebenfalls gefragt
- Aber: Architektur soll einfacher als die uns bisher bekannten Roboter-Architekturen sein („light-weight“) Agentenarchitekturen
- Beispiel: Zuno Agentenarchitektur (Ferguson, Müller, Pischel, and Wooldridge, 1997)

SIEMENS

CORPORATE TECHNOLOGY

The ZUNO Agent Architecture



SIEMENS

CORPORATE TECHNOLOGY

Die Interaktionsmanagement-Ebene

- **Aufgabe: Aktorik, Sensorik und Messaging**
- **Schnittstelle zur externen Welt realisiert durch Event Handler**
- **Service-Handler:**
  - **Routet eingehende service-bezogenen Nachrichten zu den entsprechenden Modulen**
  - **Routet ausgehende Service-Anfragen zu den entsprechenden Event Handlern**
- **Resource Manager:**
  - **Ressourcen = lokales Wissen und Informationen über andere Agenten ("aquaintances")**
  - **"Taskable" und "Non-taskable" Service-Anfragen: Non-taskable Anfragen können ohne explizite Aufgabenplanung bedient werden.**

SIEMENS

CORPORATE TECHNOLOGY

Die Aufgabenmanagement-Ebene

- Implementiert die Fähigkeit des Agenten, Dienste (Services) zu erbringen
- Aufgaben implementieren Dienste
- Aufgaben-Manager erhält Service-Anfragen
- Service-Anfragen werden durch einen dreistufigen Prozess abgewickelt:
  - Aufgabenprozessor (AP): interpretiert Aufgaben (Planbibliothek)
  - Aufgabenscheduler (AS): ordnet Aufgaben Ressourcen zu
  - Aufgabenausführung (AA): Ausführung und Monitoring von Aufgaben
- Aufgabenmanager verwendet Lastausgleich-Strategie, um neue Anforderungen einem Aufgabenprozess zuzuweisen
- Aufgaben-Meta-Controller: Koordination zwischen Aufgabenprozessen

SIEMENS

CORPORATE TECHNOLOGY

Lerninhalte dieses Kapitels

- Deliberative, reaktive und hybride Architekturen für intelligente Agenten
- Grundlegende Verfahren von Wissensrepräsentation und maschineller Planung