

# 1 Einleitung

**F1:** Was ist ein eingebettetes System?

**A1:** TO DO

**F2:** Beispiel für eingebettete SW?

- A2:**
- Manipulation der Umwelt (automatischer Straßenbau)
  - Dienstleistungen (Automatisches Tanken)
  - Aktive Heilmittel (Prothesen)

**F3:** Beispiele für (Teil-)Autonome Systeme?

- A3:**
- Unterhaltung(Aibo)
  - Unzugängliche Bereiche (Kanalisation, Sondermülllager)
  - Gefahrenzonen (AKW)

**F4:** Märkte für IES?

- A4:**
- Domestic Devices:
    - Zentrale Dienstleistungen (Energieverwaltung, Bestelldinste, Remote-Steuerung)
    - Intelligent rooms, apartments, houses (MIT Media Lab, VHE Middleware)
  - Benchmarks für künstliche Intelligenz:
    - mind (Schach)
    - mind & body (Büro Reinigung)
    - mind & body & group (Roboterfußball)

**F5:** Was ist Intelligenz?

- A5:**
- Vermögen zur Lösung von konkreten und abstrakten Problemen
  - Bewältigung von in der Erfahrung auftretenden Anforderungen und Situationen durch theoretisches Begreifen von Sinnzusammenhängen und praktische Umsetzung
  - Abstraktes Denken, Auffassungsgabe, Kreativität
  - Tierische Intelligenz: Einsichtiges Verhalten, Lösen neuer Aufgaben infolge eines Einfalls

- Intelligenz als Marketing Label: Intelligente Toaster, ...
- Künstliche Intelligenz:
  - Symbolisch: Logiken, Reasoning
  - Sub-symbolisch: neurale Netze
  - "nouvelle AI": reaktives Verhalten, embodied AI, situated AI
- Grundlagenforschung: konstruktives Verstehen von Intelligenz (AI), untrennbare Aspekte von Körper, Geist, Gesellschaft
- Anwendungsorientierte Forschung und Entwicklung: Sensor/Aktor-Schnittstellen, Autonomes Verhalten ohne menschliche Überwachung, Kommunikation, Networking

**F6:** Was ist Autonomie?

- A6:**
- auto = selbst + Nomos = Regel, Gesetz
  - Bedeutung: von menschlicher Eigenschaft bis zum Marketing (mobile Geräte)
  - Kognition: Erkennen, Wahrnehmen, Kenntnis
  - Ansatz: Selbst + Entscheidungen  $\Rightarrow$  Kognition + Leben ???
  - "Intelligenz", um -ressourcen zu bekommen und zu beaufsichtigen
  - Technischer Ansatz: Autonome Systeme, Rekursive Konstruktion von Komponenten und Methoden
  - Unabhängigkeit: Menschen hängen voneinander ab, Soziales Umfeld, Gemeinsame Strategien um ein Ziel zu erreichen, Kommunikation, Technische Nachbildung "Multi Agenten Systeme"

**F7:** Wozu braucht man Intelligenz in ES?

- A7:**
- Umwelt zeigt nichtdeterministisches/intelligentes Verhalten (z.B. andere intelligente Systeme, Menschen)
  - Daher reicht deterministisches Verhalten nicht aus
  - Es wird ein Weltmodell benötigt ("Vorstellung über die Umwelt")
  - Environment Properties:
    - accesible  $\Leftrightarrow$  inaccessible
    - deterministic  $\Leftrightarrow$  non deterministic
    - episodic  $\Leftrightarrow$  non episodic
    - static  $\Leftrightarrow$  dynamic
    - discrete  $\Leftrightarrow$  continuous

**F8:** Beispiele für Intelligenz in ES?

- A8:**
- Robotik (Industrieroboter, Servie Roboter, Personal Roboter)
  - Z.B. Pathfinder, RoboCup
  - Intelligenz in Spielzeugen: Gameboy, Tamagotchi, Storyteller, Furby, Aibo
  - Beispiele für intelligente Dienste:
    - Orientierung: Bewegung in einer Umgebung, Karte
    - Kommunikation: Sprachverarbeitung, Spracherkennung, Sprachausgabe
    - Bilderkennung: Hindernisse, Landmarken, Objekte
    - Emotionen: Ziele, Absichten, Annahmen
    - Planen: Wegeplanung, Handlungsplanung
    - Anpassung, Lernen: Einstellen auf die Umgebung, auf Situation

**F9:** Klassifikation von Robotern?

- A9:**
- Robotortypen: Einfacher Manipulator → intelligenter humanoider Roboter → "Teamwork"
  - Klassifikation:
    - Industrieroboter (wenig Autonomie, geringer Entwicklungsgrad): eindeutige Aufgabenstellung, automatengerechte Umwelt
    - Service Robot (mittel, mittel): vorgegebenes Weltmodell, automatische Bahnplanung, Einsatz in der Öffentlichkeit, multifunktionale Sensoren
    - Personal Robot (viel, hoch): Kommunikation mit der Umwelt, Verständnis der Umgebung, selbständige Überwachung, Generieren von Programmen, Lernfähigkeit

**F10:** Was ist Artificial Life and Cybernetics?

- A10:**
- Ziel: Verstehen von lebenden Systemen aus den Arbeitsprinzipien von Maschinen (und umgekehrt, Bionics)
  - ...

**F11:** Wozu Emotionen?

- A11:**
- Anzeige des internen Zustands eines Systems
  - Beeinflussen der (menschlichen) Umwelt

- Emotionen zur Steuerung von Verhalten
- Emotionen zur Steuerung von Gruppen

**F12:** Was sind Agenten, Taxonomie, Typologie?

**A12:** • **Taxonomie:**

Autonomous Agents:

1. Robotic Agents (IES)
2. Biological Agents
3. Computational Agents:
  - (a) Artificial Life Agents
  - (b) Software Agents:
    - i. Task-specific Agents
    - ii. Entertainment Agents
    - iii. Viruses

• **Typologie:**

- Collaborative Learning Agents: Corporate + Learn
- Collaborative Agents: Corporate + Autonomous
- Interface Agents: Autonomous + Learn
- Smart Agents: Corporate + Autonomous + Learn

**F13:** Problemstellungen bei IES?

- A13:** • Architekturen
- Navigation
  - Modellierung (Statecharts, DGL)
  - Kommunikation
  - Kooperation

## 2 Architekturen

**F14:** Zeitstrahl?

- A14:** • Steuerung beeinflusst E/A des "jetzt"
- Regelung beeinflusst Steuerung unter Einbeziehung von E/A aus der nahen Vergangenheit

- Lernen beeinflusst Regelung unter Einbeziehung von Daten aus der Vergangenheit. Lernen aus der Vergangenheit um in der Gegenwart besser entscheiden zu können. Erfahrungen aus der Vergangenheit müssen komprimiert/abstrahiert werden.
- Planung wird von Steuerung beeinflusst, beeinflusst selbst Regelung. Versucht in die Zukunft zu schauen (mehrere Verläufe möglich)

**F15:** Agentenarchitekturen?

**A15:** • *Bestandteile und Ablauf erklären können*

- Basic Control: für sicherheitskritische Systeme, damit sich das System z.B. nicht selbst zerstört
- **Ideal Rational Agent:**
  - Available Data:
    - \* Performance measure that defines degree of success
    - \* Everything that the agent has perceived (Folge von Wahrnehmungen)
    - \* Wissen über die Umgebung
    - \* Aktionen, die der Agent durchführen kann
  - Für jede mögliche Folge von Wahrnehmungen macht der IRA die beste Aktion (anhand der Bewertung)
  - Problem: alle möglichen Wahrnehmungsfolgen müssten bekannt und gespeichert sein.
- **Simple Reflex Agent:**
  - Weltzustand ergibt sich **nur** aus den aktuellen Sensordaten
  - Aktionsauswahl **kann** nichtdeterministisch sein (einfachste Art von Lernen)
  - Auswahl der Aktion z.B. anhand eines Gütekriteriums aber auch durch Zufall (z.B. "Capture the Flag oder Defend")
- **Reflex Agent with Internal State:**
  - *Ablauf von Zustand, Weltentwicklung, Aktionseffekte, Automatenbild?*
  - betrachte auch Vergangenheit (nur vorheriger Zustand und letzte Aktion)
  - Erzeuge Möglichkeiten über Folgezustand (schaue *nur* einen Schritt in die Zukunft)
  - Agent hat Zustand (z.B. Angriff, Verteidigung)
  - Zustandsübergang kann abhängig sein von Lernprozessen
  - Zustandsbasiert, reaktiv
  - Weltentwicklung: z.B. bringt Verteidigung was?
  - *erkläre in Verbindung mit Motorschemes*

- \* Wähle anhand des internen Zustands einen Satz von Verhalten aus, setze Parameter und Gewichte der Verhalten
- Subsumption Method: Eingaben/Ausgaben unterer Ebenen können von oberen Ebenen unterdrückt/ersetzt werden. Untere Ebenen funktionieren auch ohne die oberen Ebenen.
- **Utility Based Agent**
- **Learning Agent**
- **General Agent Architecture**

**F16:** Was sind Motorschemes?

- A16:**
- Vektorfelder, z.B:
    - Move2Goal: Vektoren zeigen zum Ziel
    - Avoid: Vektoren zeigen vom Hindernis weg
    - FollowLine: Vektoren zeigen zur Linie und in Fahrtrichtung
  - Gewünschte Schemes werden überlagert
  - Deadzones können mit zusätzlichen Rauschen verhindert werden
  - Roboter betrachtet nur Vektoren an seiner aktuellen Position und berechnet daraus den resultierende Vektor (Vektoraddition mit Normierung)
  - Normierung: verwende gewichtete Vektorsumme und grenze Anzahl der Vektorfelder ein
  - "einfaches Lernen": setze Parameter für Verhalten anhand von Erfahrungen
  - Team Bots: "Winner takes it all" ⇒ Wähle betragsmäßig größten Vektor

**F17:** Hierarchische Agentenarchitekturen?

- A17:**
- Vertical: perception, modelling, planning, task execution, motor control
  - Horizontal: avoid objects, wander, explore, build maps, monitor changes, identify objects, plan changes to the world, reasoning about behaviour of objects
  - Probleme bei hierarchischen Agenten (Sensorik vs. Logik):
    - Vertical: evtl. zur Implementierung neuer Funktionen notwendige Daten werden schon auf einer unteren Ebene herausgefiltert
    - Horizontal: Gefahr für redundanten Code sehr groß
    - Auf Entscheidungsebene (Parameterauswahl) bekommt man nicht mehr mit, ob z.B. ein Sensor A besser arbeitet als eine Sensor B.

**F18:** Triple Tower?

- A18:**
- Eingabe/Perception → Logik/Model → Ausgabe/Action
  - Ausnahme einfache Reflexe: Perception → Action
  - Hierarchie innerhalb der einzelnen Tower:
    - Perception Tower: Raw Daten, gefilterte Daten / Kantenbild, Feature
    - Model Tower: Vektorfeld → Plan
    - Action Tower: Basic Controls → ...

**F19:** Hybride Architekturen?

- A19:**
- Layer 1: Reactive Layer: Raw sensor data processing by automaton
  - Layer 2: Medium Layer: Knowledge level view of the environment, symbolic representation
  - Layer 3: Upper Level: Social aspects of the environment, goals, beliefs of other agents
  - Control System: For mediation between the different levels (rule set)
  - Bezogen auf Team Bots:
    - Level 1 (Reaktiv): Eingabe Daten, rein reaktiv "wie komme ich überhaupt zur Fahne?" (Vektoren)
    - Level 2: Symbolische Repräsentation : "Habe Fahne", "Habe Fahne nicht" "Was muss ich tun, damit ..." (Automaten)
    - Level 3: Z.B. Verteilung von Aufgaben (Strategie) "Du bist Verteiger/Angreifer aushandeln"

**F20:** Welche Auswahlmöglichkeiten für Verhalten gibt es?

**A20:** competitive, cooperative

**F21:** Unterschied Deliberative, Reactive?

- A21:**
- Reaktiv: Parallel: Modify World, Create Maps, Discover New Areas, Avoid Collisions, Move Around (Kismet)
  - Deliberativ: Sequentiell: Sense, Model, Plan, Act (Shakey)

**F22:** Hat MEXI ein Weltmodell?

- A22:**
- Nein, er hat nur einen internen Zustand (Emotion+Bedürfnisse) + Wahrnehmung. Zusätzlich Anregungsfunktionen unterhalb der Emotionen und Drives, welche Verhalten anregen.
  - MEXI = Utility Bases Agent **nur** mit "Nützlichkeit" und "Bewertung + Aktionsauswahl"
  - Look Around als Default Verhalten

**F23:** Modellierungen?

- A23:**
- Spezifische Algorithmen:
    - Orientierung
    - Sprache (Hidden-Markov Chains,...)
    - Vision (Filter, Kanten, Flächen, ...)
  - Suchen: Entscheidungsbäume
  - Logiken:
    - Präpositionale Logik (and, or, not, ...)
    - Prädikatenlogik (exist, for all)
    - Modallogiken (Temporallogiken, BDI-Logiken)
    - Unschärfe Logiken ([0,1] Logiken, z.B. Fuzzy Logik)
  - Neuronale Netze (Perzeptron, feed-xx, ...)
  - Situated Automata

### 3 Logik

**F24:** Knowledge Based Agent?

- A24:**
- Hat Knowledge Base (Anwendungsspezifisches Wissen) und Inference Engine (Anwendungsunabhängige Algorithmen)
  - Knowledge Base = Menge von Sätzen in formaler Sprache
  - Beschreibungsebenen:
    - Wissens Ebene (Agentenwissen unabhängig von Implementierung)
    - Logische Ebene (Wissen in formaler Sprache codiert)
    - Implementierungsebene (Datenstrukturen und Algorithmen zur Manipulation der WB)
  - Problemstellung:

- Formulierung des Wissens
- Ableitung (Schlussfolgerung) neuen Wissens
- Formulierung und Regeln für Ableitung neuen Wissens sollen für unterschiedliche Problemstellungen einsetzbar sein (d.h. unabhängig von der modellierten Welt)
- Ableitung neuen Wissens soll effizient automatisierbar sein

**F25:** Logiken?

- A25:**
- formale Sprachen zur Repräsentation von Wissen
  - erlauben Schlussfolgerungen zur Ableitung neuen Wissens
  - Syntax: definiert wohlgeformte Sätze (Formeln) der Sprache
  - Semantik: definiert Bedeutung der Sätze (Wahrheitswert in einer Welt)
  - Logische Konsequenz:
    - Formeln  $f$  ist logische Konsequenz aus einer Wissensbasis  $WB$  ( $WB \models f$ ) g.d.w.  $f$  in allen Welten wahr ist, in denen  $WB$  wahr ist.
  - Ableitung/Schlussfolgerung (inference):
    - Eine Formel  $f$  kann mit Hilfe einer Schlussfolgerungsregel  $i$  aus  $WB$  abgeleitet werden ( $WB \vdash_i f$ )
    - Schlussfolgerungsregel  $i$  ist konsistent (sound), wenn gilt

$$WB \vdash_i f \Rightarrow WB \models f$$

- Schlussfolgerungsregel  $i$  ist vollständig (complete), wenn gilt

$$WB \models f \Rightarrow WB \vdash_i f$$

**F26:** Propositionale Logik?

- A26:**
- Semantik:
    - Logische Konstante: *True* ist die wahre, *False* ist die falsche Aussage
    - Propositionssymbole:
      - \* Interpretation  $I$ : Abbildung der Prop. Symb. in die modellierte Welt
      - \* Wahrheitsbelegung  $B$ : Abbildung der Prop. Symb. nach  $\{True, False\}$
    - Komplexe Formeln: Wahrheitsbelegung wird auf Wahrheitstabelle für Konnektivitäten zurückgeführt
  - Modell (allgemein):
    - Welt aus Primitiven, die die jeweilige Logik erlaubt

- erlaubt die Auswertung von Wahrheitswerten (Wissenszuständen)
- $m$  ist ein Modell einer Formel  $f$ , wenn  $f$  in  $m$  wahr ist
- $M(f)$  ist die Menge aller Modelle von  $f$
- $WB \models f$ , g.d.w.  $M(WB) \subseteq M(f)$
- Propositionale Logik: Eine Interpretation  $I$ , die eine Formel  $f$  erfüllt, ist ein Modell von  $f$ .
- Im allgemeinen gibt es umso weniger Modelle je mehr Formeln eine Wissensbasis enthält
- Gültige Formel (Tautologie):
  - Eine Formel ist gültig, wenn sie in allen Modellen wahr ist (alle Reihen in der Wahrheitstabelle sind *True*).
- Erfüllbare Formel:
  - Eine Formel ist erfüllbar, wenn sie in mindestens einem Modell wahr ist (mindestens eine Reihe der Wahrheitstabelle ist *True*).
- Nicht erfüllbare Formel:
  - Eine Formel ist nicht erfüllbar, wenn sie in keinem Modell wahr ist (alle Reihen der Wahrheitstabelle sind *False*).
- Gültigkeit – Schlussfolgerung:
  - $WB \models f$  g.d.w.  $(WB \Rightarrow f)$  ist gültig
- Erfüllbarkeit – Schlussfolgerung:
  - $WB \models f$  g.d.w.  $(WB \wedge \neg f)$  ist nicht erfüllbar
  - gebräuchliche Beweistechnik für  $f$ : Reduktion ad absurdum, d.h. Annahme  $(WB \wedge \neg f)$  sei wahr und Herleitung eines Widerspruchs (Reduktion auf leere Menge)
- Wahrheitstabellen – Eigenschaften:
  - Der Test auf Gültigkeit/Erfüllbarkeit einer Formel ist
    - konsistent** - es werden nur gültige/erfüllbare Formeln als gültig/erfüllbar ermittelt
    - vollständig** - alle gültigen/erfüllbaren Formeln werden als gültig/erfüllbar ermittelt
  - Der Aufwand für die Prüfung einer Menge von propositionalen Formeln auf Erfüllbarkeit ist **exponentiell**
  - Allgemeines Resultat (Cooke): Die Prüfung einer Menge von propositionalen Formeln auf Erfüllbarkeit ist NP-vollständig
- Schlussfolgerungsregeln zur Ableitung von Wissen:

**Modus Ponens**

$$\frac{V \Rightarrow S, V}{S}$$

**Und-Eliminierung**

$$\frac{P_1 \wedge P_2 \wedge \dots \wedge P_n}{P_i}, 1 \leq i \leq n$$

**Und-Einführung**

$$\frac{P_1, P_2, \dots, P_n}{P_1 \wedge P_2 \wedge \dots \wedge P_n}$$

**Oder-Einführung**

$$\frac{P_i}{P_1 \vee P_2 \vee \dots \vee P_n}, 1 \leq i \leq n$$

**Eliminierung doppelter Negation**

$$\frac{\neg\neg P}{P}$$

**Unit Resolution**

$$\frac{P \vee Q, \neg Q}{P}$$

**F27:** Prädikatenlogik?

**A27:** • Semantik:

- Sätze sind wahr bezüglich einer Welt und einer Interpretation  $I$
- Welt beinhaltet Objekte und Relationen zwischen ihnen
- Interpretation  $I$ :
  - \* Konstantensymbole  $\rightarrow$  Objekte
  - \* Prädikatsymbole  $\rightarrow$  Relationen
  - \* Funktionsymbole  $\rightarrow$  funktionale Relationen
  - \* Variablensymbole haben keine feste Interpretation, sie können während der Auswertung an verschiedene Objekte gebunden werden
- Wahrheitsbelegung  $B$ :
  - \* bildet atomare/komplexe Formeln nach  $\{True, False\}$  ab
  - \* Atomare Formeln: Prüfung, ob n-Tupel von Terminiinterpretationen Element der  $P$  entsprechenden Relation ist
  - \* Komplexe Formeln: Wahrheitsbelegung analog zur Propositionalen Logik durch iterative Anwendung der Wahrheitstabelle
  - \* Komplexe Formel  $\forall xP$ : Konjunktion
  - \* Komplexe Formel  $\exists xP$ : Disjunktion
- Schlussfolgerung – Beweis
  - Beweis einer Formel  $f$ :
    - \* Konsistente Schlussfolgerung:  $WB \models f$
    - \* Suchprozess über Formeln mit Hilfe von Schlussfolgerungsregeln

\* z.B. Modus Ponens (MP):

$$\frac{P \Rightarrow Q, P}{Q}$$

\* z.B. Und-Einführung (AI):

$$\frac{P, Q}{P \wedge Q}$$

\* z.B. Universal Eliminierung (UE):

$$\frac{\forall x P}{P\{x/T\}}$$

– Unifikation: Idee: Unifiziere Voraussetzungen mit bekannten Fakten, wende Unifier (Substitution  $\sigma$ ) auf Schlussfolgerung an.

– Verallgemeinerter Modus Ponens (GMP):

$$\frac{p'_1, p'_2, \dots, p'_n, (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\sigma} \text{ mit } p'_i\sigma = p_i\sigma \text{ für alle } i$$

Der GMP ist konsistent aber nicht vollständig. Außerdem setzt er eine bestimmte Form der *WB* voraus (Horn Klauseln)

• Ableitung neuen Wissens durch Resolution:

– Weitere Schlussfolgerungsregeln:

\* Unit Resolution:

$$\frac{P \vee Q, \neg Q}{P}$$

\* Resolution:

$$\frac{P \vee Q, \neg Q \vee R}{P \vee R} \text{ bzw. } \frac{\neg P \Rightarrow Q, Q \Rightarrow R}{\neg P \Rightarrow R}$$

– Resolution einer Formel durch Widerspruch: um  $WB \models P$  zu zeigen, zeige, dass  $WB \wedge \neg P$  nicht erfüllbar ist

– Voraussetzung: *WB* und *P* in KNF

– Inferenz durch Resolution verwendet 2 Klauseln, um eine neue zu erzeugen, solange, bis leere Klausel abgeleitet ist (Widerspruch)

– KNF Umformung:

1. Ersetze Implikation

2. Verschiebe  $\neg$  nach innen

3. Umbenennen von Variablen, so dass jeder Quantor eigene Variable

4. Eliminieren von  $\exists$  mittels Skolem Konstante (außerhalb von  $\forall$ ) / Funktion (innerhalb von  $\forall$ )

5. Linksverschieben der Quantoren

6. Weglassen der Quantoren

7. Distributivgesetz anwenden

– Probleme beim Resolutionsbeweis: Auswahl der nächsten Klausel, der zu unifizierenden Literale

- Lösungen:
  - \* Vereinfachung der Klauselmenge  $K$ : Löschen von tautologischen Klauseln, löschen einer Klausel, die von einer anderen Klausel aus  $K$  subsumiert wird
  - \* Vollständige Suchstrategien für die Auswahl der Klauseln:
    - Breitensuche**
    - Set of Support** – Idee: zielgerichtete Klauselauswahl
      - wähle Formelmenge i.allg. negierte Zielformel als set of support
      - jeder Schritt kombiniert Element aus SOS mit einer anderen Klausel und fügt Resolvente in SOS ein
    - Unit preference** – Idee: kürzere Formeln erzeugen
      - bevorzuge Resolutionen, wo eine Klausel sogenannte unit clause ist, d.h. nur aus einem Literal besteht
    - Linear resolution** – Idee:  $P$  und  $Q$  können resolviert werden, wenn
      - $P$  ist in Originalwissensbasis oder
      - $P$  ist Vorgängervon  $Q$  im Beweisbaum
- ABER: Prädikatenlogik 1. Ordnung ist semientscheidbar, d.h.
  - \* Beweis einer Formel  $P$  wird gefunden, wenn  $WB \models P$
  - \*  $WB \not\models P$  ist nicht immer beweisbar (unendliche Laufzeit möglich)

Resolution und alle anderen vollständigen und konsistenten Inferenzmethoden für PL 1. Ord. sind NP-hart. Lösungsmöglichkeit: Einschränkung der Syntax (z.B. Prolog)
- Horn Klauseln – PROLOG:
  - \*  $p_h : -p_1, \dots, p_n$  entspricht  $p_1 \wedge \dots \wedge p_n \Rightarrow p_h$
  - \*  $p_h$  heißt Kopf (head) der Klausel
  - \*  $p_1, \dots, p_n$  heißt Körper (body) der Klausel
  - \* Fakten (facts):  $p_h : -$
  - \* Ziele (goals):  $: -p_1, \dots, p_n$
  - \* Typischer Aufbau von Prolog Programmen: Ziele, Fakten, Regeln
  - \* Ableitung neuen Wissens/Beweis durch backward chaining mit Tiefensuche unter closed world assumption (d.h. wenn eine Prädikat nicht bewiesen werden kann, dann wird seine Negation als wahr angesehen) (da Tiefensuche ist Inferenz unvollständig)
  - \* Backward Chaining:
    - Ausgangspunkt: Ziel  $q$
    - Beweis( $q$ ):
      - Fakt  $q'$ , falls  $q$  mit  $q'$  unifiziert werden kann
      - falls  $q$  mit dem Kopf  $q'$  einer Klausel  $k$  unifiziert werden kann mache alle Literale von  $k$  zu neuen Zielen und versuche sie durch backward chaining zu beweisen

## 4 Planung

**F28:** Was ist Planung?

- A28:**
- Erstelle Vorstellung, wie sich die Zukunft entwickeln könnte
  - Dazu wird Weltmodell benötigt
  - Planen kann man auch Lernen
  - Hinweis: Schachcomputer  $\Rightarrow$  keine richtige Planung, da Zeitaufwand vernachlässigt wird.
  - Da in der Zukunft mehrere Verläufe möglich sind und Planung nach jedem Schritt neu stattfindet, ist Planung oft nicht in Echtzeit möglich
  - Branch and Bound Problem
  - Versuche daher möglichst viel reaktiv zu gestalten

**F29:** Planungssystem allgemein?

- A29:**
- Symbolic model of the environment (subset of first order logic)
  - Symbolic specification of the actions (PDA: precondition, add, delete)
  - Planning Algorithm:
    - input: representation of the environment
    - input: set of action specifications
    - input: representation of al goal state
    - output: plan for execution
    - Ablauf: while TRUE do
      - \* pick a goal
      - \* generate a plan
      - \* execute a plan
  - Too slow for real world applications

**F30:** Planungsaufgabe?

- A30:**
- Planungsaufgabe:
    - Finde ausgehend vom Startzustand eine Aktionsfolge (Plan), der einen Zielzustand / eine Menge von Zielzuständen erreicht.
    - u.U. sind "nebenbei" Kosten zu minimieren

**F31:** Was ist die Grundlage (Problembeschreibung) für das Planen?

- A31:**
- Startzustand
  - Operatoren = Zustandsänderung / Effekte jeder Aktion
  - Zielbeschreibung z.B als
    - Menge von Zielzuständen oder
    - Funktion, die entscheidet, ob Zustand ein Zielzustand ist
    - logische Formel
  - Pfadkosten (optional)

**F32:** Was ist der Vorteil von Planen?

**A32:** Aktionen müssen erst ausgeführt werden, wenn ihre Simulation mittels der Operatoren sie als sicher und effektiv ausgewiesen hat.

**F33:** Charakterisierung von Planungsproblemen?

- A33:**
- Welt deterministisch, vollständig beobachtbar
    - ⇒ Effekte von Aktionen beobachtbar
    - ⇒ 1 Folgezustand nach Aktion (z.B. deterministische Automaten)
  - Welt deterministisch, nicht vollständig beobachtbar
    - ⇒ Effekte von Aktionen nicht vollständig beobachtbar
    - ⇒ Menge von möglichen Folgezuständen nach Aktion (z.B. nichtdeterministische Automaten)
  - Welt nichtdeterministisch, nicht vollständig beobachtbar (tatsächliche Aktionsfolge verursacht andere Effekte als durch Operatoren beschrieben / Welt verändert sich)
    - ⇒ Anpassung des Plans erforderlich (z.B. verzahnte Planung und Ausführung)

**F34:** Was ist Planen durch Suchen?

- A34:**
- Zustände: Datenstrukturen (Knoten im Suchgraph)
  - Aktionen: Code für Expansion / Operatorfunktion
  - Ziel: Code
  - Plan: Zustandsfolge (Aktionsfolge)
  - Beispiele: Spiele (Schach, Dame, ...), Wegeplanung, Roboternavigation (mit teilweise bekannten Hindernissen), Montageroboter

**F35:** Allgemeine Realisierung von Suchalgorithmen?

- A35:**
- dynamische Erzeugung des Suchbaums/-graphen durch Expansion von Knoten
  - Suchstrategie bestimmt Expansionsreihenfolge
  - i.allg. Verwendung einer Suchliste zum Festhalten der Reihenfolge, d.h. Einfüge/Ausfüge-  
modus bestimmt die Suchstrategie
  - Expansion eines Knoten N:
    - Knoten entsprechen Zuständen
    - Ermittlung der Nachfolgezustände von N (mittels Operatoren auf zugehörigem  
Zustand von N)
    - Erzeugen von Nachfolgeknoten aus Nachfolgezuständen

**F36:** Bewertung von Suchstrategien?

- A36:**
- Bewertung:
    - Vollständigkeit – wird vorhandene Lösung immer gefunden
    - Zeitkomplexität – Anzahl von erzeugten/expandierten Knoten
    - Speicherkomplexität – maximale Anzahl von Knoten im Speicher
    - Optimalität – wird immer die Lösung mit geringsten Kosten gefunden
  - Charakteristische Größen:
    - b – maximaler Verzweigungsfaktor
    - d – Tiefe der gefundenen Lösung
    - m – maximale Tiefe des Zustandsraums (möglicherweise unendlich)

**F37:** Suchstrategien?

**A37:** • Uninformierte Suche:

	Zeit	Speicher	Vollständig	Optimal
Breitensuche	$O(b^d)$	$O(b^d)$	ja, $b < \infty$	ja
Tiefensuche	$O(b^d)$	$O(bm)$	nein, Schleifen, $m = \infty$	nein
Tiefenlimitierte S	$O(b^l)$	$O(bl)$	ja, für $l \geq d$	nein
”Iterative deepening”	$O(b^d)$	$O(bd)$	ja	ja
Bidirektionale Suche	$O(b^{d/2})$	$O(b^{d/2})$	ja	ja

- Informierte Suche:
  - Kostenminimierung (Best First):
    - \* Greedy (kleinstes  $h(n)$ , geschätzte Kosten von  $n$  bis zum Ziel):

- (wie Tiefensuche, erst Pfad in die Tiefe bis zum Ende/Misserfolg absuchen, dann Backtracking)
- nicht vollständig, nicht optimal
- \* A\* ( $f(n) = g(n) + h(n)$ ;  $g(n)$  tatsächl. Kosten bis  $n$ ,  $h(n)$  gesch. Kosten bis zum Ziel):
  - vollständig, optimal, optimal effizient
  - Speicherverbrauch exponentiell für fast alle praktischen Heuristiken; nur wenn  $|h(n) - h^*(n)| \leq O(\log h^*(n))$ , dann subexponentiell
- Speicheroptimierung:
  - \* Iterative deepening A\* (IDA\*) (depth first mit Kostenlimit  $l$  für jede Iteration):
    - vollständig, optimal
    - Speicherbedarf, gute Schätzung:  $bd$
    - Zeitproblem für komplexe Domänen, z.B. TSM
  - \* Simplified Memory Bounded A\* (SMA\*) (genau soviel Speicher nutzen wie verfügbar):
    - vollständig, wenn genug Speicher für die kürzeste Lösung
    - optimal, wenn genug Speicher für die optimale Lösung
    - optimal effizient, wenn genug Speicher für gesamten Suchraum
- Iterative Verbesserung:
  - \* Hill-climbing (Evaluierungsfunktion eines Zustands = Höhe, Startzustand zufällig, Folgezustand: ein Zustand mit größerer Höhe):
    - nicht optimal, kann in lokalen Minima stecken bleiben oder zufällig auf Plateaus herumirren  $\rightarrow$  random restart hill climbing
    - Speicherbedarf: aktueller Zustand
  - \* Simulated annealing (Folgezustand zufällig, geringere Höhe erlaubt mit best. Wahrscheinlichkeit  $e^{\Delta E/T}$  (Verschlechterung  $\Delta E$  und Temperatur  $T$  sinken mit jeder Iteration)):
    - optimal, wenn  $T$  langsam genug sinkt
    - Speicherbedarf: aktueller Zustand
- Sonstige:
  - \* Island-Driven Search (Festlegung von Knoten  $(n_1, \dots, n_k)$  durch die gute Suchpfade führen, Zerlegung der Suche in  $k$  Teilsuchen von  $n_{\text{start}}$  nach  $n_1$ ,  $n_1$  nach  $n_2$ ,  $\dots$ ,  $n_k$  nach  $n_{\text{goal}}$ ):
    - nicht optimal, nicht vollständig
    - Speicherbedarf: abhängig von geringerer Knotenzahl

**F38:** Welche Heuristiken gibt es?

**A38:**

**F39:** Was ist Logisches Planen?

- A39:**
- Zustände: Logische Formeln
  - Aktionen: Vorbedingungen (Preconditions), Effekte
  - Ziel: Logische Formel
  - Plan: Constraints auf Aktionen

**F40:** Planen als Theorembeweis?

- A40:**
- z.B. Situation Calculus (Prädikatenlogik 1. Stufe)
  - Zu jedem Prädikat, das nicht "ewig" wahr ist, wird ein Situations- oder Zustandsargument hinzugefügt (z.B.  $On(x, y, s)$ ,  $s$ =Situation(Zustand))
  - Situationen/Zustände werden durch die *Result* Funktion verbunden ( $Result(a, s)$  ist der Folgezustand, der aus  $s$  nach Aktion  $a$  resultiert)
  - Effekt-Axiome (beschreiben Änderungen durch Aktionen)  
 $\forall s[On(x, y, s), \dots \Rightarrow On(x, z, Result(move(x, y, z), s))]$
  - Frame-Axiome (beschreiben "Nicht-Änderungen" durch Aktionen)  
 $\forall s[(On(x, y, s) \text{ und } x \neq u) \Rightarrow On(x, y, Result(move(u, v, z), s))]$
  - Frame Problem: Beschreibung aller "Nicht-Änderungen" nötig  
 $\Rightarrow$  viele Frame-Axiome  
 $\Rightarrow$  wiederholtes Kopieren zur Zustandsaktualisierung
  - Planen im Situation Calculus:
    - Gesucht: Aktionsfolge
    - Antwort als Zustandsfolge, die durch Aktionen erzeugt wird
  - Festlegung von Regeln zur Benutzung von Prädikatenlogik
  - Planen = allgemeine Resolution zur Ableitung von Wissen
  - $\Rightarrow$  Planungssysteme erfüllen diese Aufgabe jedoch viel effizienter.

**F41:** Zustandsraumsuche vs. Planungsraumsuche?

- A41:**
- Zustandsraumsuche: Speicherung / Aneinanderreihung von Zuständen

- Planungsraumsuche:
  - Plan besteht aus Steps, teilweise mit Precondition
  - Planer wählt Precondition aus und fügt Step, der Precondition als Effekt hat dem Plan hinzu

**F42:** Planen als Zustandsraumsuche (total order)?

- A42:**
- Vorteil: keine Frame-Axiome, unverändertes wird nicht beachtet
  - z.B. STRIPS:
    - States: Menge von positiven Literalen (z.B. On(B,A))
    - Operators:
      - \* operator: Name, Argumente
      - \* precondition: Konjunktion positiver Literale
      - \* delete list: Literale, die aus dem Weltzustand gelöscht werden
      - \* add list: Literale, die zum Weltzustand zugefügt werden
  - Plan (Pfad durch Suchbaum) finden mit:
    - Progressionssuche (Vorwärts):
      - \* startet mit initialem Zustand
      - \* Auswahl von Aktionsfolgen und Prüfung, ob das Ziel im resultierenden Zustand erfüllt ist
      - \* ohne Heuristiken impraktikabel in realistischen Anwendungen
    - Regressionssuche (Rückwärts) (in STRIPS):
      - \* startet am Zielzustand
      - \* bildet Zielformel zurück, um Subziele zu erhalten
      - \* bildet iterativ Subziele zurück, bis das so erreichte Subziel durch den aktuellen Zustand erfüllt ist
      - \* Regression einer Formel  $f$  durch Regel (Operator)  $r$  ist die schwächste Formel  $f'$  ( $f'$  ist schwächer als  $f''$ , wenn  $f'' \models f'$ )
      - \* z.B.  $P$  ist schwächer als  $P$  und  $Q$ ,  $P$  oder  $Q$  ist schwächer als  $P$

**F43:** Planen als Planungsraumsuche (partial order)?

- A43:**
- Planungsraumsuche:
    - Knoten = partieller Plan
    - Operatoren = Verfeinerung / Modifikation partieller Pläne

- Startet mit unvollständigem Plan (partieller Plan) und verfeinert / modifiziert ihn z.B. durch das Einfügen von Planungsschritten, bis ein vollständiger Plan erreicht ist.
- Verfeinerung:
  - schränkt initialen unvollständigen Plan immer mehr ein, bis ein vollständiger, korrekter Plan erreicht ist.
- Modifikation:
  - erlaubt auch andere Änderungen als Verfeinerungen, z.B. wenn initial auch inkorrekte Pläne erzeugt werden
- Partieller Plan:
  - Menge von Planungsschritten (steps, problemspez. Operatoren)
  - Menge von Ordnungsbedingungen zwischen steps ( $S_i < S_j$ ), i.allg. partielle Ordnung
  - Menge von Variablenbindungen ( $v = x$ ,  $x$ : Variable oder Konstante)
  - Menge von Kausalbeziehungen (causal links) ( $S_i \rightarrow c S_j$ ,  $S_i$  erreicht Bedingung  $c$  für  $S_j$ )
- Verfeinerungsoperatoren:
  - add a link (von Aktion zu offener Bedingung)
  - add a step (zur Erfüllung einer offenen Bedingung)
  - order a step wrt another
  - add a variable binding
- Offene Bedingung:
  - nicht erfüllte Vorbedingung (precondition) eines Planungsschrittes
- POP, TOP *erklären*
- *Beispiel erklären können*

**F44:** Was sind Clobbers?

- A44:**
- Kausale Links im Plan sind geschützt
  - Threats oder Clobbers sind mögliche Schritte, die solch einen Link zerstören könnten, wenn sie zwischendurch ausgeführt würden.
  - Lösung: Demotion (zuerst), Promotion (danach)

**F45:** Partial Order Planning?

- A45:**
- Konjunktive Planung basierend auf causal links

- Eigenschaften von POP:
  - vollständig
  - konsistent
  - systematisch (keine Wiederholungen)
- Problem: Für Pläne mit vielen Schritten ist der Suchraum für POP riesig  $\Rightarrow$  Hierarchisches Planen

**F46:** Hierarchisches Planen?

- A46:**
- Operatoren: primitive Operatoren, abstrakte Operatoren
  - Dekomposition abstrakter Operatoren (Makrodefinition, Subroutine)
  - Plan  $p$  implementiert Operator  $o$ , wenn gilt:
    - $P$  ist ein vollständiger und konsistenter Plan für die Erreichung der Effekte von  $o$  aus den Preconditions von  $o$
    - $\Rightarrow p$  ist konsistent: kein Widerspruch bei der Ordnung der Constraints für Variablenbindungen von  $p$
    - $\Rightarrow$  Jeder Effekt von  $o$  muss von mindestens einem Schritt von  $p$  erfüllt werden (und darf von keinem späteren verletzt werden)
    - $\Rightarrow$  Jede Precondition in  $p$  muss entweder durch einen Schritt in  $p$  erreicht werden oder eine Precondition von  $o$  sein.
  - Mögliche Threats zwischen Subschritten müssen beachtet werden.
  - Effizienzsteigerung durch Hierarchisches Planen benötigt Upward/Downward Solution Property
  - Upward Solution Property:
    - Wenn ein abstrakter Plan  $p$  inkonsistent ist, dann gibt es keine primitive Lösung von der  $p$  eine Abstraktion ist.
  - Downward Solution Property:
    - Wenn  $p$  eine abstrakte Lösung ist, dann gibt es eine primitive Lösung, von der  $p$  eine Abstraktion ist.
  - Beachte: Upward/Downward Solution Property sind keine logische Folge aus konsistenter/vollständiger Hierarchie. Müssen separat gefordert sein.
  - z.B: unique main subaction condition:
    - Alle Preconditions und Effekte eines abstrakten Operators werden einem Schritt des Plans auf der nächst niedrigeren Hierarchieebene zugewiesen  $\Rightarrow$  garantiert upward solution property.
  - Beispiel:

- Nicht hierarchische Planung:  $O(b^n)$
- Hierarchische Planung:  $\sum_{i=1}^d bs^i = O(bs^d)$

**F47:** Probleme beim Planen?

**A47:** • Problem:

Reale Welt verhält sich nicht immer wie geplant, z.B. wegen

- unvorhersehbarer Ereignisse (Reifen platzt)
- unvollständige Information (Reifen noch intakt?)
- inkorrekte Information (fehlerhafte Sensordaten, Effekt einer Aktion unvollständig)

Qualification Problem: Man kann nicht alle möglichen Vorbedingungen und alle möglichen bedingten Effekte einer Aktion auflisten

• Lösung:

– Conditional Planning:

In die Planung werden sogenannte Beobachtungsaktionen einbezogen, um fehlende Information zu beschaffen.

⇒ Teilplan für jede Eventualität und Ausnahmebedingung

⇒ sehr ineffektiv, da viele unwahrscheinliche Fälle

– Monitoring und Replanning:

Die Planung berücksichtigt die normalen Zustände und Effekte. Während der Ausführung wird der Fortschritt beobachtet und falls nötig (Abweichungen von ursprünglichen Annahmen) eine erneute Planung angestoßen.

⇒ unerwartete (nicht berücksichtigte) Effekte führen zu Fehlern, auf die nicht adäquat reagiert werden kann

**F48:** Conditional Planning?

**A48:** • Wie POP, ausser: Wenn eine offene Bedingung durch eine Beobachtungsaktion geprüft werden kann, dann

- füge Beobachtungsaktion (BA) in den Plan ein
- vervollständige den Plan für alle möglichen Ergebnisse von BA
- jedes mögliche Ergebnis bestimmt einen sogenannten Kontext
- füge bedingten Schritt (consitional step) mit diesen Teilplänen und entsprechendem Kontext ein

**F49:** Monitoring und Replanning?

- A49:**
- Ausführungsüberwachung: Fehler, wenn Preconditions des verbleibenden Plans verletzt
  - Aktionsüberwachung: Fehler, wenn Preconditions der nächsten Aktion verletzt (oder Aktion selbst misslingt)
  - Einfaches Replanning:
    - gesamten Plan  $g$  für gegebenes Ziel berechnen
    - Für jeden Zustandswechsel (Perception - Action - Cycle) prüfen, ob Preconditions  $p$  des Plans  $g$  erfüllt
    - falls nein:
      - \* wähle besten Teilplan  $t$  aus, der von aktuellem Zustand zum Ziel führt
      - \*  $p' =$  Preconditions von  $t$
      - \* berechne Plan  $s$  für Subziele  $p'$
      - \* neuen Gesamtplan  $g$  zusammensetzen aus  $[s, t]$
    - erste geplante Aktion ausführen und aus  $g$  entfernen
    - $\Rightarrow$  Planung und Ausführung sind zwei getrennte Prozesse. Für jede Wahrnehmung muss neuer Gesamtplan berechnet werden, auch wenn sich nach der nächsten Aktion wieder alles ändert.
  - Integrierte Planung und Ausführung:
    - Planung und Ausführung werden verzahnt
    - Ausführung kann schon mit partiellem Plan starten (insb. für unabhängige Teilziele)
    - Restliche Planung berücksichtigt dann schon Wahrnehmungen während dieser Teilausführung
      - $\Rightarrow$  weniger Initialaufwand
      - $\Rightarrow$  aktuellere Planung

**F50:** Graphbasiertes Planen (Disjunktives Planen)?

- A50:**
- Planungsgraph: gerichteter Ebenengraph besteht aus zwei Ebenen, die sich "abwechseln"
  - 2 Arten von Knoten:
    - Propositionale Knoten
    - Aktionsknoten
  - 3 Arten von Kanten:
    - precondition Kanten

- add Kanten
- delete Kanten
- Planungsvorgehen:
  - Start: Planungsgraph PG mit einer Propositionsebene für Startbedingungen
  - Ebene  $i$ :
    - \* erweitere PG der Ebene  $i-1$  um einen Zeitschritt (Aktionsebene+Propositionseben)
    - \* Suche einen gültigen Plan der Länge  $i$  (backward chaining),
    - \* wenn gefunden, dann anhalten, wenn Ziele nicht erfüllbar zu Zeit  $i$ , gehe zu Ebene  $i + 1$
- Mutual exclusion (mutex) relations:
  - 2 Aktionen auf Ebene  $i$  sind mutual exklusiv, wenn kein gültiger Plan beide beinhalten kann.
  - 2 Propositionen auf Ebene  $i$  sind mutual exklusiv, wenn kein gültiger Plan beide wahr machen kann
- Mutual exklusiv Propositionen auf einer Ebene können sich auf einer späteren Ebene nicht mehr ausschließen
- Propagierung von mutual exklusiv relations:
  - Aktionen:
    - \* Interferenz: Wenn eine Aktion eine Precondition/Add-Effekt einer anderen löscht
    - \* Konflikte: Wenn eine Precondition  $p$  einer Aktion  $a$  und eine Precondition  $q$  einer Aktion  $b$  existieren, die auf der vorangehenden Graphenebene als mutually exklusiv gekennzeichnet wurden.
  - Propositionen:
    - \* Wenn bei zwei Propositionen  $p$  und  $q$  auf einer Ebene sind und alle Wege zur Erreichung von  $p$  die Erreichung von  $q$  ausschließen
    - \* Wenn jede Aktion  $a$ , die  $p$  als Add-Effekt hat mutually exklusiv ist zu jeder Aktion  $b$ , die  $q$  als Add-Effekt hat.

## 5 Modallogik

**F51:** BDI Agenten?

- A51:**
- BDI Theorie: Verständnis des praktischen Schlussfolgerns
  - Theoretisches Schlussfolgern: Herleitung von Wissen
  - Praktisches Schlussfolgern:

- Aktionsgerichtet
- Was soll erreicht werden?
- Wie soll es erreicht werden?
- Beliefs: Informationen über die Welt (u.U. unvollständig, unkorrekt)
- Desires: Wünsche über zukünftige Gegebenheiten (u.U. widersprüchlich)
- Intentions "committed desires": Resultat der Deliberation (widerspruchsfrei)
- Intentions:
  - "state of mind", der zu Aktion führt
  - sind bestätigt (Intentionen werden nicht grundlos revidiert)
  - beeinflussen means-ends reasoning, Agent versucht Intentionen zu erreichen
  - schränken zukünftige delibertain ein, inkonsistente Optionen (mit Intentionen) werden ausgeschlossen
  - beeinflussen beliefs für zukünftiges praktisches Schlussfolgern, Erreichbarkeit der Intention wird geglaubt
- Simple BDI Agent Algorithm:
  - while true do
  - B:=belief-revision(B,p) \*aktualisiert Informationen über Welt
  - D:=options(B,I) \*erzeugt Alternativen
  - I:=filter(B,D,I) \*wählt aus konkurrierenden Alternativen
  - P:=plan(B,I) \*erzeugt/wählt Plan um I zu erreichen
  - execute(P)
  - end while

**F52:** BDI Modell?

- A52:**
- Einsatzbereich:
    - Multiagentensysteme
    - Kommunikation zwischen Agenten, erfordert den Austausch von Wissen/Glauben und Intentionen
    - Koordination von Agenten, efordert Kenntnisse der Intention anderer Agenten
  - Anwendungscharakteristika:
    - Umgebung: nicht deterministisch, nicht beobachtbar, dynamisch
    - viele Ziele, die sich gegenseitig beeinflussen
    - jederzeit verschiedene Aktionen möglich
    - beste Aktionsauswahl von Umgebung abhängig

- BDI Logik:
  - Formalisierung eines BDI Modells
  - Grundlegende Aussagen über Eigenschaften eines Modells (Verhalten der Agenten)
  - Basiert auf Modallogik (z.B. LORA)

**F53:** Modallogik?

- A53:**
- Notwendige Aussagen ( $N$ ): sind in allen möglichen Welten wahr
  - Kontingente Aussagen ( $M$ ): Für eine kontingente Aussage wäre eine Welt vorstellbar, in der die Aussage nicht gilt
  - Aussagen  $A = U \cup M$ ,  $N \subseteq W \subseteq M$  ( $U$ = Unmögliche Aussagen,  $W$ =Wirklichkeit)
  - Kontingente Aussagen sind weder notwendig noch unmöglich
  - $M$  beinhaltet auch widersprüchliche Aussagen
  - Mögliche Aussagen sind in mindestens einer möglichen Welt wahr
  - Die Wirklichkeit entspricht einer der möglichen Welten
  - Jede mögliche Welt beinhaltet nurwiderspruchsfreie Aussagen
  - Notwendige Aussagen sind in allen möglichen Welten wahr
  - Syntax:
    - $\Box\alpha$ :  $\alpha$  ist notwendig (Agent  $i$  weiß/glaubt  $\alpha$ )
    - $\Diamond\alpha$ :  $\alpha$  ist möglich, es gilt  $\Diamond\alpha \Leftrightarrow \neg\Box\neg\alpha$
  - Semantik:
    - wahrheitsfunktionale Interpretation von  $\Box\alpha/\Diamond\alpha$  nicht möglich  $\Rightarrow$  Interpretation mit Hilfe möglicher Welten
    - Wahrheit von Aussagen bezieht sich auf Frames  $F = (W, R)$  mit  $W$ =Menge von möglichen Welten,  $R \subseteq W \times W$ =Erreichbarkeitsrelation, d.h. ein Frame gibt für jede "aktuelle Welt" an, welche möglichen Welten von dort aus erreichbar sind
    - Modell: Ein Modell für eine propositionale Modallogik ist ein Tripel  $m = (W, R, \pi)$  mit einer Wahrheitsbelegung  $\pi : W \times Prop \rightarrow \{True, False\}$
    - Wahrheit in einer Welt bzgl. eines Modells  $m = (W, R, \pi)$ :
      - \*  $(m, w) \models \alpha$ :  $\alpha$  ist wahr/erfüllt in der Welt  $w$  bzgl. des Modells  $m$
      - \* Für atomare propositionale Formeln  $\alpha$  gilt:  $\pi(w, \alpha) = True$  oder  $\pi(w, \alpha) = False$ ,  $(m, w) \models \alpha$  g.d.w.  $\pi(w, \alpha) = True$
      - \*  $(m, w) \models True$
      - \*  $(m, w) \models \neg\alpha$  g.d.w.  $(m, w) \not\models \alpha$
      - \*  $(m, w) \models \alpha \vee \beta$  g.d.w.  $(m, w) \models \alpha$  oder  $(m, w) \models \beta$

- \*  $(m, w) \models \alpha \wedge \beta$  g.d.w.  $(m, w) \models \alpha$  und  $(m, w) \models \beta$
- \*  $(m, w) \models \Box\alpha$  g.d.w.  $(m, u) \models \alpha$  für alle  $u$  mit  $(w, u) \in R$
- \*  $(m, w) \models \Diamond\alpha$  g.d.w.  $(m, u) \models \alpha$  für ein  $u$  mit  $(w, u) \in R$
- Frames über die Zeit:
  - \*  $W$  – Situationen über der (linearen) Zeit, eine Welt  $w(t)$  entspricht den Gegebenheiten zum Zeitpunkt  $t$
  - \*  $R$  – von  $w(t)$  sind alle zukünftigen Welten  $w(t')$  erreichbar mit  $t' \geq t$
  - \*  $R$  ist reflexiv, transitiv, seriell (KTD4 System)
- Frames über einen Zustandsraum (Bsp.):
  - \*  $W$  – Mögliche Kartenverteilungen bei 3 Skatspielern
  - \*  $R$  – von einer Kartenverteilung/Welt  $w$  sind alle Kartenverteilungen/Welten  $w'$  erreichbar, die nach den Regeln erlaubt sind
  - \*  $R$  ist reflexiv, transitiv aber  $N$  gilt nicht (aus A hat P As folgt nicht  $\Box(A$  hat P As))
- Erfüllbarkeit:
  - Eine Formel  $\alpha$  heißt erfüllbar in einem Modell  $m$ , wenn es eine Welt  $w$  in  $m$  gibt mit  $(m, w) \models \alpha$
- Gültigkeit in einem Frame ( $F \models \alpha$ ):
  - Eine Formel  $\alpha$  heißt gültig in einem Frame  $F = (W, R)$  oder  $F$ -gültig, wenn sie in allen Modellen  $m = (W, R, \pi)$  die auf  $F$  basieren, in allen Welten  $w \in W$  wahr ist
- Gültigkeit in einer Klasse von Frames:
  - Eine Formel  $\alpha$  heißt gültig in einer Klasse von Frames  $C$  oder  $C$ -gültig, wenn sie in allen Frames  $F \in C$  gültig ist. ( $K$  bezeichnet die Klasse aller Frames)
- Bsp. für  $K$ -gültige Formeln:
  - $\alpha \vee \neg\alpha$
  - $\Box(\alpha \vee \neg\alpha)$
  - $\Box\alpha$ , falls  $\alpha$  eine Tautologie ist
  - $K : \Box(\alpha \Rightarrow \beta) \Rightarrow (\Box\alpha \Rightarrow \Box\beta)$  wird auch Axiom/Schema  $k$  genannt
- Normale Modallogische Systeme:
  - $K$  –  $\Box(\alpha \Rightarrow \beta) \Rightarrow (\Box\alpha \Rightarrow \Box\beta)$
  - $N$  –  $\frac{\alpha}{\Box\alpha}$  Notwendigkeitsregel
  - $MP$  –  $\frac{\alpha, \alpha \Rightarrow \beta}{\beta}$  Modus Ponoens
  - $US$  –  $\frac{\alpha}{\alpha\sigma}$  Uniform Substitution
  - $\sigma$  ersetzt einige oder alle Variablen  $v_1, \dots, v_n$  durch zulässige modallogische Formeln  $\Phi_1, \dots, \Phi_n$

- Problem der logischen Allwissenheit:
  - In Systemen mit Axiom  $K$  und Notwendigkeitsregel besitzt eine Agent logische Allwissenheit, d.h.
    - (a) er weiß alle gültigen Formeln, u.a. alle propositionalen Tautologien (folgt direkt aus der Notwendigkeitsregel)
    - (b) sein Wissen ist abgeschlossen unter der Implikation (Sei  $\beta$  logische Konsequenz von  $\alpha_1, \dots, \alpha_n$ , dann muss in jeder Welt, in der alle  $\alpha_1, \dots, \alpha_n$  wahr sind, auch  $\beta$  wahr sein, und daher ist auch  $\alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta$  eine gültige Formeln. Wegen  $N$  muss der Agent diese Formel auch wissen und wegen  $K$  muss er auch  $\beta$  wissen)
- Axiome – Bedeutung:
  - $T \ \Box\alpha \Rightarrow \alpha$  **Knowledge Axiom** Was gewusst wird ist wahr (Zur Unterscheidung zw. Wissen und Glauben).  
Erreichbarkeitsrelation: reflexiv –  $\forall w \in W, (w, w) \in R$
  - $4 \ \Box\alpha \Rightarrow \Box\Box\alpha$  **Positive Introspection Axiom** Ein Agent weiß was er weiß.  
Erreichbarkeitsrelation: transitiv –  $\forall w, u, v \in W, (w, u) \in R \wedge (u, v) \in R \Rightarrow (w, v) \in R$
  - $5 \ \Diamond\alpha \Rightarrow \Box\Diamond\alpha$  **Negative Introspection Axiom** Ein Agent weiß was er nicht weiß.  
Erreichbarkeitsrelation: euklidisch –  $\forall w, u, v \in W, (w, u) \in R \wedge (w, v) \in R \Rightarrow (u, v) \in R$
  - $D \ \Box\alpha \Rightarrow \Diamond\alpha$  Ein Agent weiß/glaubt nichts widersprüchliches.  
Erreichbarkeitsrelation: seriell –  $\forall w \in W \exists u \in W, (w, u) \in R$
  - S5 (KT5) üblich als Logik des Wissens
  - Schwaches S5 (KD45) üblich als Logik des Glaubens
  - KT heißt auch T-System
  - KT4 heißt S4-System
  - KT5 heißt S5-System
  - KD45 heißt schwaches S5-System
- LORA:
  - Grundlage Prädikatenlogik 1. Ordnung
  - Belief–Desire–Intention Komponente
  - Temporale Komponente: zur Formulierung der Weiterentwicklung über die Zeit
  - Aktionskomponente: zur Repräsentation von Aktionen eines Agenten und ihrer Effekte
  - Beliefs:
    - \* Bel  $i \phi$  – Agent  $i$  glaubt  $\phi$
    - \* Bel  $(i(\text{Bel } j \phi))$  – Schachtelung erlaubt: Ag.  $i$  glaubt, dass  $j \phi$  glaubt.

- \* Erreichbarkeitsrelation ist seriell, transitiv und euklidisch (logic of beliefs ist einer KD45 Logik (Schwaches S5))
- Desires, Intentions:
  - \* Syntax wie beliefs
  - \* Erreichbarkeitsrelation ist seriell (logic of desire/intention ist eine KD Logik)
- Temporale Komponente:
  - \* Grundlage: Branching Time Structure
  - \* Pfad Konnektivitäten in LORA:
    - $\bigcirc\phi$   $\phi$  ist im nächsten Zeitpunkt/Zustand wahr
    - $\diamond\phi$   $\phi$  ist eventuell wahr in der Zukunft
    - $\square\phi$   $\phi$  ist immer wahr in der Zukunft
    - $\phi U \Phi$   $\Phi$  ist irgendwann in der Zukunft wahr und solange ist  $\phi$  wahr
    - $\phi W \Phi$  wir  $U$ , aber es kann sein, dass  $\Phi$  nie erfüllt wird
  - \* Pfad Quantifizierung in LORA:
    - $A\phi$   $\phi$  ist auf allen Pfaden wahr
    - $E\phi$   $\phi$  ist auf mindestens einem Pfad wahr
- Aktionskomponente:
  - \* Operatoren zur Repräsentation von Aktionen:
    - (Happens  $\alpha$ )** Aktionsausdruck  $\alpha$  findet als nächstes statt
    - (Achvs  $\alpha \phi$ )** Aktion  $\alpha$  findet statt und erreicht  $\phi$
    - (Agts  $\alpha g$ )** Gruppe  $g$  soll Aktion  $\alpha$  ausführen
  - \* Aktionsausdrücke:
    - $\alpha; \alpha'$  Sequenz
    - $\alpha | \alpha'$  Alternative
    - $\alpha^*$  Wiederholung, 0 oder mehrmals
    - $\phi?$   $\phi$  ist erfüllt (z.B.  $\phi?; \alpha | \neg\phi? \alpha'$ )
- Semantik von LORA:
  - \* wird mit Hilfe möglicher Welten definiert (possible world semantics)
  - \* Welten sind branching time structures, d.h. es wird nicht nur Unsicherheit über den aktuellen Zustand der Welt sondern auch über zukünftige Entwicklungen repräsentiert
  - \* Erreichbarkeitsrelationen geben Erreichbarkeit einer Welt  $u$  aus einer Welt  $w$  (branching time structur) zu einem bestimmten Zeitpunkt  $t$  an
  - \* Erreichbarkeitsralation für beliefs ist seriell, transitiv und euklidisch (logic of beliefs ist eine KD45 Logik (Schwaches S5))
  - \* Erreichbarkeitsrelationen für desire/intention sind seriell (logic of desire/intention ist eine KD Logik)

- Einsatz von LORA:
  - \* formale Charakterisierung einiger Typen rationaler Agenten
  - \* Untersuchung von Beziehungen zwischen beliefs, desires auf Basis der Erreichbarkeitsrelationen
- Beispiel:
  - \* Realismus:  $(\text{Bel } i \phi) \Rightarrow (\text{Des } i \phi)$  bzw.  $(\text{Bel } i \phi) \Rightarrow (\text{Int } i \phi)$
  - \* Ein Agent wünscht (intendiert) das was er glaubt
  - \* für jede desire (intention)-accessible Welt  $w$  zu einem Zeitpunkt  $t$  gibt es eine belief-accessible Welt, von der  $w$  eine Teilwelt ist
- Mögliche Einsatzbereiche: Spezifikation, Implementierung, Verifikation

## 6 Perzeption

**F54:** Image processing issues?

- A54:**
- Many to one
  - changing over time
  - 2-dim array of values (RGB, YUV)
  - iconic model
  - Scene based model
  - noisy pictures
  - extracting information using specific knowledge
  - navigation
  - manipulation
  - predict probable future changes

**F55:** Was sind Probleme bei Perzeption?

- A55:**
- Schatten
  - unterschiedliche Farben
  - Merkmale mit unterschiedlicher Bedeutung
  - Bild hängt nicht nur von Kamera sondern auch von Umwelt ab
  - Kameras haben sehr viel geringere Dynamik als menschliches Auge  $\Rightarrow$  Nachregeln erforderlich
  - depth discontinuity (Kante zu Hintergrund)

- surface reflectance discontinuity
- surface normal discontinuity (Kante zu Kante)
- illumination discontinuity (Schatten)
- Beleuchtung
- Bewegungsunschärfe
- innere Kalibrierung (Verzerrung)

**F56:** Grundlegende Schritte für Bildverarbeitung?

**A56:** • Image Processing:

- concerned with the image as an image (Eingabe: Pixelbild  $\Rightarrow$  Datenmenge riesig)
- Verarbeitung: z.B. Filtern, Kanten hervorheben, Regionen finden
- Ausgabe sind **keine** Pixelmuster, sondern komprimierte Darstellung (z.B. charakterisierte Kanten)

• Scene analysis:

- attempt to infer properties of the world
- create an iconic description
- create a feature based description
- providing task specific information
- Ausgabe: Liste von Merkmalen
- kann evtl. aufgabenbezogen die Parameter der Bildverarbeitung anpassen (oder auch Kameraautomatik abschalten)

**F57:** Digitale Bildverarbeitung?

**A57:** • Bildaufnahme  $\rightarrow$  Pixelbilder

• Bildvorverarbeitung  $\rightarrow$  Pixelbilder

- Pattern Matching  $\rightarrow$  Objekte in 2D. Auch Eingabe für Stereo Analyse und/oder Tracking
- Stereo Analyse und/oder Tracking  $\rightarrow$  Tiefe
- Merkmalsextraktion** Pixelbilder (Farbraum, Auflösung, Framerate)  $\rightarrow$  Vektorisierte Merkmale (Farbregionen, Kanten, Linien, Ecken, ...)

**Objekterkennung**  $\rightarrow$  Objekte in 2D

**3D-Berechnung mittels Modell**  $\rightarrow$  Objekte in 3D

• funktionale Beschreibung

**F58:** Rauschen eliminieren?

**A58:** • Averaging mittels Faltung:

$$1D \quad s^*(t) := \int s(u)w(u-t)du = s(t) * w(t)$$

$$2D \quad I^*(x, y) = I(x, y) * W(x, y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} I(u, v)W(u-x, v-y)$$

• Gauß:

$$W(x, y) = G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

**F59:** Kantenverstärkung?

**A59:** • 1. Ableitung = Kantenstärke

• 2. Ableitung mit Nulldurchgang (zero-crossing)  $\Rightarrow$  Kante

• In Kombination mit Gauß Smoothing (1D:  $I^*(x) = I(x) * G(x)$ ):

$$d^2[I^*(x)]/dx = d^2[I(x) * G(x)]/dx^2 = I(x) * d^2G(x)/dx^2$$

• Laplace (Kanten) in Kombination mit Gauß  $\Rightarrow$  Mexican Hat

• Sobel Filter:

**Horizontaler Filterkern**

$$h_{horizontal} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

**Vertikaler Filterkern**

$$h_{vertical} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

**Richtungswerte** 2 pro Pixel (x,y):

– magnitude:  $\sqrt{x^2 + y^2}$  oder  $|x| + |y|$

– direction:  $\tan^{-1}(y/x)$  bzw. Rundung mittels "Stoppschild"

• Beispiel PDA-AR:

– Low Level Vision: Original  $\rightarrow$  Sobel  $\rightarrow$  Non Maxima Eliminierung  $\rightarrow$  Skelettierung  $\rightarrow$  Vektorisierung

– 1. Phase: Vektorisierte Kanten

– 2. Phase: Verlängerung der Kanten, Schnittpunkte bilden Knoten des Graphen, Kantengraph

– 3. Phase Typisierung aufgrund von Form und Anordnung im Graphen, Typisierung der Nachbarkanten

**F60:** Feature extraction?

- A60:**
- Eingabe: Pixelbild (ca. 75000 pixel)
  - Parallel: "Edge detection & Tracking" und "Color Segmentation & Tracking" → ca. 100 features
  - Reduktion per Feature selection auf ca. 20 features
  - nearl static dataflow
  - usage of synchronous dataflow graphs appropriate
  - slice based approach: z.B. one image line, 8x8 pixel block ⇒ geringere Latenz (Verarbeitung während Aufnahme)

**F61:** Scene Analysis?

- A61:**
- Kanten Beschriften: occlude edges(→), convex edge (+), concave edge (-)
  - junctions Beschriften: V junction, W junction, Y junction, T junction
  - labeled scene: problem searching for a consistent labeling (not unique) constraint problem
  - scene model using generalized cylinders: e.g. A,B,C and 6 location parameters

**F62:** Entfernungsmessen?

- A62:**
- Monobilder:  $d = h / \tan \alpha$
  - Statische Stereobilder:  $d = b / (\tan \alpha + \tan \beta)$
  - Dynamische Stereobilder: Nacheinander aufgenommene Bilder aus unterschiedlichen Positionen

**F63:** Region Finding?

- A63:**
- Kriterien für Regionen: Abweichungen in einer Region, Durchschnittswerte für eine Region
  - Eigenschaften:
    - Helligkeit**
    - Farbe** Oranger Ball, Farbcodierungen

**Texturen, Reflexionen** Tiere (Fell), Gras, Bäume (Rinde, Blätter)

- Finde Regionen in 2D Bild erfordert Ähnlichkeitskriterium (ÄK)
- Probleme:
  - Einfacher Vergleich notwendig, da ÄK auf jedes Pixel angewandt wird
  - Aus Performancegründen heißt "rechteckiger Bereich (Würfel) ist ähnlich"
  - "Wolke" in YUV wird auch durch Rechteck angenähert
  - Beachte: Wird ÄK nur auf Nachbarn angewandt, dann kann u.U. das ganz Bild in einer Region sein (Farbverlauf, Regenbogen)
  - Fangfrage: Kann man den gleichen Vergleich in RGB und YUV machen? Machen ja, aber es werden in der Regel andere Farben erkannt da immer ein Würfel gewählt wird.
- Verfahren nach Horowitz:
  - Teile, Vergleiche, Merge
  - Nachteil: Man benötigt das ganze Bild, man kann nicht mit einer Zeile anfangen
  - Vorteil: Beliebige Homogenitätskriterien möglich

**F64:** Dirks Algorithmus?

**A64:** • Idee:

- Input: camera image → no info of the colors of the objects
- camera/grabber: YUV values → use YUV values most of the time
- fast → go through the image line-by-line
- recognition of colored regions → skip non-colored pixels
- size, position and color of the objects → representation of the regions using moments
- Moments:
  - $M(p, q) = \sum_{(x,y) \in region} x^p y^q$
  - $M(0, 0)$  = number of pixels of the region
  - $M(1, 0)$  = Sum of all X coordinates
  - $M(0, 1)$  = Sum of all Y coordinates
  - $c_x = \frac{M(1,0)}{M(0,0)}$  center of the region
  - $c_y = \frac{M(0,1)}{M(0,0)}$
- Central Moments:
  - $C(p, q) = \sum_{(x,y) \in region} (x - c_x)^p (y - c_y)^q$
  - $width = \sqrt{\frac{C(2,0)}{M(0,0)}}$

- $height = \sqrt{\frac{C(0,2)}{M(0,0)}}$
- $diagonal = \sqrt{\frac{C(1,1)}{M(0,0)}}$
- moments of second order  $\Rightarrow$  description of an ellipse
- Data structure of a region: moments of second order, mean YUV value (insg. 6 Zahlen für eine Region:  $M(0,0), C(2,0), C(1,1), C(0,2), c_x, c_y$ )
- Algorithmus:
  - Erst Zeile für Zeile Line Regions erstellen (based on YUV, hier YUV gut, da Änderung von einem Pixel zum nächsten meistens sehr gering)
  - Mergen der Regionen:
    - \* Berechnung von einer Verschmelzung in konstanter Zeit
    - \* Hier funktioniert HSI Modell besser, da Farbunterschied von einem Moment zum nächsten im YUV Modell zu weit auseinander liegen könnte (man müsste einen zu großen Bereich erlauben). In HSI hätte z.B. der Ball im idealen Fall überall den gleichen H Wert, auch wenn der Ball oben überbelichtet und unten im Schatten ist.
    - \* Line Regions sind in Liste nach aufsteigender Y Koordinate gespeichert
    - \* Durchlaufe diese Liste und Merge zwei Regionen, wenn sich umgebende Rechtecke (approximieren Ellipse, z.B. mit Achsenparallelen Rechtecken) schneiden (oder nur um einen Offset verschoben sind)
    - \* Worst Case  $O(n^2)$  (jede Region muss mit jeder verglichen werden) wird nur selten erreicht, da wenn sich einige Regionen (Parameter) sich schon nicht mit der zu überprüfenden Region schneiden, dann werden dies auch die folgenden in der Liste nicht tun (da ja nach aufsteigender Y Koordinate gespeichert)
- Objekterkennung:
  - algorithm searches for two (marker three) colored objects
  - colors, sizes and shapes of the objects are given
  - compares color, size-ratio and shapes
  - distance can be calculated
- Vorteile:
  - no specification of colors to search
  - image will be analyzed linearly
  - no need for sharp
  - more fault-tolerant than edge extraction
  - small artifacts have little influence on results
  - liefert direkt Richtungsvektor mit (Vektor auf Schwerpunkt)
  - Reduktion des Datenvolumens
  - Tracking möglich

- Nachteile:
  - only suitable for simple shapes
  - less exact than edge extraction
  - thin vertical regions will not be detected
  - run-time for region merging
  - alot of parameters

**F65:** Tracking?

- A65:**
- Feature Kriterien: Farbe ähnlich, Form der Ellipse ähnlich, Schwerpunkt ähnlich
  - erfordert zeitlich sehr nahe Bilder (Problem, wenn sich Kamera selbst bewegt), beachte Abtasttheorem
  - VisiTrack:
    - Feature Extraction (75000 pixels  $\rightarrow$   $\approx$  20 features)
    - Point Selection
    - Feature 3D position calculation ( $\rightarrow$   $\approx$  20 features)
    - Kalman Filter ( $\rightarrow$   $\approx$  20 features)
    - Camera Position Calculation ( $\rightarrow$   $\approx$  20 features)
  - Vorteile von Tracking mit Momenten:
    - gleich Vektor mit Richtung (z.B. für Motor-Schemes)
    - auch wichtig für Verhalten: z.B. MEXI bewegt Augen und Hals, so dass Winkel zu interessantem Objekt =  $0^\circ$
    - Verhalten durch Tracking ohne explizite Ausprogrammierung
    - z.B. CamFollowBall beim Paderkicker

**F66:** Inkrementelles Tracking?

- A66:**
- Notwendig um 3D Koordinaten von einem Objekt zu berechnen
  - Stereobasis über die Zeit
  - Feature darf sich nicht bewegen

**F67:** Real Time 3D Camera Tracking?

**A67:**

**F68:** Pattern Matching?

**A68:** • Beispiel: Person finden

1. Bildaufnahme
2. Farbsegmentierung zur Vorauswahl
3. Suche mittels Kopf-Pattern
4. Personenvermessung (Stereobild erforderlich)
5. Personenidentifikation

## 7 Sensor Fusion

**F69:** Deterministische Modelle?

**A69:** • Vorgehen

- Mathematisches Modell (Korrespondenz Modell, Regelstrecke)
  - \* Erfahrung, physikalische Gesetze, empirisches Testen, ...
- Konstruktion eines Reglers
  - \* Generierung der passenden Eingaben, um die gewünschten Ausgaben zu erhalten
- Überwachung von Systemgrößen
  - \* Konstruktion von Sensoren, Vorverarbeitungseinheiten
- Nachteile:
  - Mathematische Systeme sind nicht perfekt (nur kritische Werte werden erfasst, Vernachlässigung von Effekten, da sonst zu umfangreich)
  - Störgrößen werden vernachlässigt (Störgrößen sind nicht kontrollierbar, Störgrößen können nicht deterministisch modelliert werden)
  - Sensoren erzeugen keine genauen Daten (Nicht alle Größen können gemessen werden, Sensordaten enthalten Messfehler)
  - (Schätzung aufgrund von redundante (Sensor-)daten)

**F70:** Stochastische Modelle?

**A70:** • Problemstellungen:

- Wie können Unsicherheiten bei der Systemmodellierung berücksichtigt werden
- Wie können verrauschte Daten optimal abgeschätzt werden
- Wie können Unsicherheiten in einer Steuerung berücksichtigt werden

- Wie können stochastische Kontrollsysteme (mit Schätzungen) evaluiert werden
- Minimieren von Fehlern

**F71:** Kalman Filter?

- A71:**
- Ziel: Optimale lineare Schätzung
  - Alle Messungen/Genauigkeiten könne verarbeitet werden
  - Berücksichtigung von Messfehlern und Unsicherheiten des dynamischen Modells
  - Berücksichtigung der Initialwerte
  - Rekursiver Datenverarbeitungsalgorithmus
  - Eingaben: Messung, Genauigkeit (ggf. konstant)
  - Ausgabe: Schätzung
  - hat Zustand (ggf. Zeitabhängig)
  - Voraussetzungen:
    - lineares Modell
    - Weißes Rauschen (Rauschen ist unabhängig von der Zeit)
    - Bandpass, Verhalten realer Systeme (rosa Rauschen = real, es gibt keine unendlich große/kleine Frequenzen)
    - Rauschen ist "breiter" als der Bandpass
    - Weißes Rauschen über dem Systembereich (Bandpass)
    - Fehler in der PG (zu viele gelbe Tore)  $\Rightarrow$  Monte Carlo Map:
      - \* Z.B. Fehler durch "Kinder mit gelben TShirts links" und "gelbes Tor rechts" kann nicht mit Gaußverteilung modelliert werden (ergäbe zwei Häufungen bezogen auf den Winkel)
      - \* Kalman Filter würde fälschlicherweise mitteln (erst gefiltert, dann stochastisch geglättet), Monte Carlo glättet erst stochastisch ("baut 2 Haufen auf") und filtert dann (über Logik)
  - Gaußverteilung:
    - System- und Messrauschen wird durch viele kleine Größen verursacht
    - Die Summe von unabhängigen Zufallsgrößen kann gut durch eine Gaußverteilung angenähert werden
    - Gaußverteilung ist vollständig durch  $z$  und  $\sigma$  bestimmt
    - $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-z)^2}{2\sigma^2}}$
    - Erwartungswert:  $EX := z$
    - Varianz:  $DX := \sigma^2$
    - Beispiel:

- \* Verteilung der Position basierend auf  $z_1$

$$f_{x(t_1)|z(t_1)}(x|z_1) = \frac{1}{\sqrt{2\pi}\sigma_{z_1}} e^{-\frac{(x-z_1)^2}{2\sigma_{z_1}^2}}$$

- \* Beste Schätzung  $\hat{x}(t_1) = z_1$
- \* Genauigkeit (Varianz)  $\sigma_x(t_1) = \sigma_{z_1}^2$
- \* Verteilung der Position basierend auf  $z_2$  zum gleichem Zeitpunkt  $t_1 = t_2$ :

$$f_{x(t_2)|z(t_2)}(x|z_2)$$

- \* Berechne Verteilung der Position basierend auf  $z_1$  und  $z_2$ 
  - Neuer Mittelwert:

$$\mu = \left[ \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] z_1 + \left[ \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] z_2$$

- Neue Varianz:

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2}$$

oder mit  $\hat{x}(t_2) = \mu$

$$\hat{x}(t_2) = \left[ \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] z_1 + \left[ \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] z_2 = z_1 + \left[ \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] [z_2 - z_1]$$

Mit  $K(t_2) = \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}$  gilt:

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)]$$

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - K(t_2)\sigma_x^2(t_1)$$

- \* Beispiel inkrementelle Positionsbestimmung:
  - Bewegung in x Richtung mit Geschwindigkeit  $u$  und Ungenauigkeit  $w$ , d.h.  $\frac{dx}{dt} = u + w$  ( $w$  wird durch Gaußverteilung mit Mittelwert 0 und Varianz  $\sigma_w$  modelliert)
  - $\hat{x}(t_3^-)$  = beste Schätzung vor der Messung der Position zum Zeitpunkt  $t_3$
  - Verarbeitung der Messung  $t_3$  mit  $z_3$  und  $\sigma_{z_3}^2$

$$\hat{x}(t_3^-) = \hat{x}(t_3^-) + K(t_3)[z_3 - \hat{x}(t_3^-)]\sigma_x^2(t_3) = \sigma_x^2(t_3^-) - K(t_3)\sigma_x^2(t_3^-)$$

- mit  $K(t_3) = \frac{\sigma_x^2(t_3^-)}{\sigma_x^2(t_3^-) + \sigma_{z_3}^2}$

- Ungenaue Messung

$$\Rightarrow \sigma_{z_3}^2 \rightarrow \infty$$

$$\Rightarrow K(t_3) \rightarrow 0$$

$$\Rightarrow \hat{x}(t_3) \rightarrow \hat{x}(t_3^-)$$

- Ungenaue inkrementelle Positionsbestimmung

$$\Rightarrow \sigma_w^2 \rightarrow \infty$$

$$\Rightarrow \sigma_x(t_3^-) \rightarrow \infty$$

$$\Rightarrow K(t_3) \rightarrow 1$$

$$\Rightarrow \hat{x}(t_3) = \hat{x}(t_3^-) + 1 \star [z_3 - \hat{x}(t_3^-)] = z_3$$

· Neue Messung nicht notwendig

$$\Rightarrow \sigma_x^2(t_3^-) \rightarrow 0$$

$$\Rightarrow K(t_3) \rightarrow 0$$

## 8 Navigation

**F72:** Was ist Navigation?

- A72:**
- Wissenschaft den Kurs eines mobilen Roboters zu bestimmen, während dieser sich in einer Umgebung (Land, Wasser oder Luft) bewegt
  - Navigationsproblem: Finde Pfad von Start zu Ziel und vermeide Kollisionen
  - Aufgaben: mapping und modelling der Umgebung, Pfad Planung und Auswahl, Pfad folgen und Kollisionen vermeiden
  - Navigation basierend z.B. auf Neuralen Netzen, Expert Systems, Rule Systems
  - Die meisten Algorithmen brechen ab, wenn sie auf Situationen stoßen, welche die Navigation erschweren

**F73:** Navigationsprozess?

**A73: Measurement**

- Sense the environment
- Detect objects
- Odometry
- Input user command

**Modelling**

- Map the environment
- Extract features
- Model Objects
- Map pathes

**Perception**

- Find pathes
- Detect collisions situations
- Learn the map

## Planning

- Decompose task into subgoals
- Select a path
- Choose alternatives when a path is blocked

## Action

- Navigate
- Traverse path and avoid collision
- Control of robot (kinematic and dynamic models)

### F74: Orientierung?

- A74:**
- Vorverarbeitung (Korrelation der Sensorik):
    - Lagekopplung von Fahrzeugmodell und Fehlermodell (z.B. mittels Kalman Filter) und Lagesensorik (z.B. Kreisel)  $\Rightarrow$  Fahrzeuglage
    - Lagekopplung dient auch als Eingabe für Lagestützung
  - Korrelation mit der Umgebung:
    - Lagestützung anhand von Landmarken und Abstandsmesung
  - Daten aus Lagestützung werden im Umweltmodell verwendet, hier auch Einbeziehung der Karte

### F75: Dead Reckoning (Reckoning=Standortbestimmung)?

- A75:**
- Durch Messfehler entstehen bei inkrementeller Positionsbestimmung (z.B. mittels Inkrementalgeber) sowohl Winkel- als auch Translationsfehler
  - Problem: Bei kleinem Schritt ist Fehler zwar klein aber bei längerer Fahrt summiert sich diese auf.
  - ("Folgendes ist zwar richtig, aber nach Bernd zu weit gesponnen")
  - Je weiter Ziel entfernt, desto größer die Abweichung bei Fahrt zum Ziel
  - Lösung:
    - Besser kalibrieren
    - Versuche Marken auf dem Pfad zu erkennen und fahre diese ab
    - Benutze Landmarken
    - $\Rightarrow$  Position/Winkel des Roboters kann bei Erreichen von Marken/Erblicken von Landmarken neu bestimmt werden  $\Rightarrow$  Fehler wird klein gehalten



**F81:** Maps based on Feature Detection?

- A81:**
- Feature is a high contrast point in the scene
  - Feature was determined using triangulation from different positions
  - Feature was stored as a circle
  - Visible objects were modelled as spheres
  - The radius of the sphere was a function of the uncertainty of the location of the feature

**F82:** Free Space Maps?

- A82:**
- Spacial Graph:
    - Nodes are stop points where the robot senses the environment
    - The arcs record the straight line motion of the robot
  - Voronoi diagram:
    - Divide environment into region, each of which is associated with a given point
    - The region associated with a point is the locus of points closer to that point than any other given point
    - Generalized Voronoi diagram:
      - \* Points which are equidistant from object boundaries
      - \* Placing a given point at the centre of each object
      - \* From triangle by connecting these points
      - \* Edges of the triangles are bisected with orthogonal lines
      - \* These lines form the edges of the voronoi diagram
      - \* Precondition: convex objects
    - Delaunay triangulation of the Voronoi diagram = dual graph of the voronoi diagram

**F83:** Topological Map?

- A83:**
- Two levels of space representation
    1. Topologic: places are nodes, connectore are arcs
    2. Geometric: dimensions of the elements of the connectivity graph, objects, walls and spaces are represented by convex polygons

**F84:** Object-oriented Maps?

**A84:**   • Start  $\rightarrow$  Object A description (vertex points or configuration vector)  $\rightarrow$  Object B description

**F85:** Composite-space Maps?

**A85:**   • Gitter  
      • z.B. Quadtree mit Knotentyp: frei, Objekt

**F86:** Rule-based Maps?

**A86:**   • Rule based model  
      • Collection of predicate calculus statements  
      • Axiom set or planning system  
      • Object classes, primitive predicates and relations  
      • z.B. Rooms (type(r1,room),name(r1,main room),grid(r1,g1)), Doors (type(d1,door), name(d1,office door),...),Faces(),Objects())

**F87:** Path Planning?

**A87:**   • Find a path through a mapped environment so that the robot can travel along it without colliding with anything  
      • handle uncertainty in the sensed world model and errors in path execution  
      • minimize the impact of objects in the field of view of the robot's sensors by keeping the robot away from those objects  
      • find the optimum path, if that path is to be negotiated regularly  
      • Suchen über Path planning graph

**F88:** Automatic Path Planning?

**A88:**   • Abstract the search space to a graph of possible paths (Spatial graph)  
      • Find the shortest path  
      • Natural for learning environments

**F89:** Free Space Planner?

- A89:**
- Voronoi diagram
  - Deal with the free space rather than with the space occupied by obstacles
  - Reduce path planning to graph search techniques
  - Pathes pass through the midpoint of the faces of connecting clear regions
  - In Verbindung mit Expanding Objects:
    - Verbinde alle Vertexe der expandierten Objekte mit Dreiecken
    - Weg führt jeweils durch Mittelpunkt der Seiten eines Dreiecks

**F90:** Free Space Maps?

- A90:**
- The boundaries of an object voronoi diagram form a network of possible pathes
  - Pathes that are to narrow must be eliminated  $\Rightarrow$  Piano mover problem
  - Lösung: Expand Objects
    - Attach a coordinate to the reference point
    - Flip the robot about the two axes of the coordinate frame
    - Place frame of the flipped robot at each of the vertices of the object
    - Find the convex hull

**F91:** Vertex Graph path through a set of expanding objects?

- A91:**
- Überlagere Slices mit Pfaden, je einen für jede Fahrtrichtung ( $A \rightarrow B$ ,  $B \rightarrow A$ )

**F92:** Composite Space Planner?

- A92:**
- Plan path from the center of one free space grid square to the center of connected free space grid square
  - Mögliche Gitter: Orthogonal 4-connected grid, Area grid, 8-connected grid

**F93:** Path Smoothing?

- A93:**
- Path relaxation:
    1. select grid pathes
    2. get off the grid
  - Cost function for pathes:

- A cost of one unit for path length
- An object cost that is an inverse function of the distance to nearby objects
- A cost for being in an unmapped, restricted or dangerous region

**F94:** Distance Transform Propagation?

- A94:**
- Benutze 8-connected distance transform (orthogonal and diagonal distances have a value of 1)
  - "Wellen" benutzen: Welle von Ziel (Wert 1) aus, bis Welle Start erreicht, dann gehe immer einen Schritt in Richtung fallender Wert

## 9 Lernen

**F95:** Lernender Agent?

**A95: Wertungskritik**

**Lernelement**

**Problem Generator**

**Performance Element (Agent)**

**F96:** Design of learning elements?

- A96:**
- Affect of learning elements:
    - Welche Komponenten des Performance elements sollen verbessert werden?
    - Welche Repräsentation wird für diese Komponenten benutzt?
    - Welches Feedback ist verfügbar?
    - Welche vorherigen Informationen (aus der Vergangenheit) sind verfügbar?
  - Komponenten eines Performance elements, welche gelernt werden können:
    - Direkte Abbildung von condition-action rules
    - Leite relevante Eigenschaften der Welt aus der Perzeptionssequenz ab
    - Wie sich die Welt entwickelt
    - Ergebnisse möglicher Aktionen
    - Nützlichkeitsinformationen über die Erwünschtheit (desirability) der Welt
    - Action-values für die Auswahl von Aktionen in bestimmten Zuständen
    - Ziele, deren Erreichen die Nützlichkeit des Agenten maximiert

- Jede Komponente kann dabei als mathematische Funktion beschrieben werden

**F97:** Verfügbares Feedback?

- A97:**
- Supervised learning: Eingabe und Ausgabe einer Komponente können beobachtet werden (Beispiele  $x, f(x)$ )
  - Reinforcement learning (reward and punishment): Agent erhält eine Bewertung (gut/schlecht) seiner Aktionen
  - Unsupervised learning: Keine zusätzlichen Informationen über die Eingabe, Lerne Beziehungen zwischen den Wahrnehmungen

**F98:** Inductive Learning?

- A98:**
- Eingabe: Menge von Beispielen  $(x, f(x))$
  - Ausgabe: Eine Funktion  $h$  (hypothesis), die  $f$  approximiert
  - Die Bevorzugung einer hypothesis gegenüber anderer wird bias genannt
  - Inkrementelles Lernen
  - Ausdrucksstark und effizient

**F99:** Learning Decision Trees (Learning from example)?

- A99:**
- Ziel: Erstelle Entscheidungsbaum, so dass alle Beispiele aus Trainig set korrekt klassifiziert werden. Dazu muss eine Reihenfolge für die Auswertung der Attribute festgelegt werden.
  - Problem: Unterscheidung von "guten" und "schlechten" Attributen:
    - Gutes Attribut:** Auswahl des Attributs führt dazu, dass möglichst viele Beispiele sofort klassifiziert werden können
    - Schlechtes Attribut:** Auswahl des Attributs führt zu Aufteilung der Beispielmengen, so dass Teilmengen nicht sofort klassifiziert werden können.
  - Ockham's raror: The most likely hypothesis is the simplest one that is consistent with all observations
  - Ablauf:
    - erstes Attribute teilt Beispiele auf
    - Jedes Kind ist ein neuer (kleinerer) decision tree
      - \* Falls negative und positive Beispiele enthalten, wähle gutes Attribut und teile weiter auf

- \* Falls alle Beispiele positiv (negativ) sind, gebe ja (nein) zurück
- \* Falls keine Beispiele mehr übrig, gebe Majorität der Klassifizierungen der Eltern zurück
- \* Falls keine Attribute mehr vorhanden, nicht genug Information (gebe Majorität zurück)
- Methode:
  - Sammle große Menge von Beispielen
  - Teile Menge auf in Training set und Test set (disjunkt)
  - Benutze Lernalgorithmus mit Training set
  - Messe prozentualen Anteil der korrekt klassifizierten Test set Beispiele
  - Wiederhole dies mit unterschiedlichen Größen des Training sets

**F100:** Learning General Logical Description?

- A100:**
- Inductive Learning:
    - Suche eine gute Hypothese im großen durch die representation language definierten Raum
  - Problem: Find an logical equivalent expression to classify examples correctly
  - Stelle Decision tree logisch dar ( $\forall r \Leftrightarrow Patrons(r, some) \vee \dots$ )
  - hypothesis space  $H = \{H_1, \dots, H_n\}$
  - Learning algorithm believes the sentence  $H_1 \vee \dots \vee H_n$
  - $D_i(X_i)$  gibt die Beschreibung eines Beispiels an ( $Alternate(X_1) \wedge \neg Bar(X_1) \wedge \dots$ )
  - Complete training set is the conjunction of all these sentences
  - Ein Beispiel ist false negative für Hypothese, falls die Hypothese besagt, das das Beispiel negativ sein müsste, dieses aber defacto positiv ist (false positiv analog)  $\Rightarrow$  Hypothese ist logisch inkonsistent mit Beispiel
  - z.B.: Hypothesis  $H_1 \vee H_2 \vee H_3 \vee H_4$ , Beispiel  $I_1$  ist inkonsistent mit  $H_2$  und  $H_3 \Rightarrow$  Leite zu  $H_1 \vee H_4$  mit inference system ab

**F101:** Current best hypothesis search?

- A101:**
- Maintain a single hypothesis and adjust it as new examples arrive (maintain consistency)
  - alle Beispiele mit Hypothese  $H$  konsistent  $\Rightarrow$  OK
  - Beispiel ist false negative  $\Rightarrow$  generalization

- Beispiel ist false positive  $\Rightarrow$  specialization

**F102:** Least Commitment Search?

- A102:**
- Idee: Halte all die Hypothesen (Version space  $V$ ) aufrecht, welche konsistent sind mit allen bisherigen Beispielen/Daten
  - VERSION\_SPACE\_UPDATE:  $V \leftarrow \{h \in V : h \text{ is consistent with } e\}$
  - Problem: Representation des großen hypothesis space
  - Lösungsidee:
    - partial ordering on the hypothesis space by generalization/specialization
    - using an intervall representation
    - each boundary will be a set of hypothesis (boundary set)
    - the current version space is the set of hypotheses consistent with all the examples so far
    - representing the version space using two boundary set
      1. G-set: a most general boundary, initially TRUE  
Every member of the G-set consistent with all observations so far and there are no consistent hypothesis that are more general
      2. S-set: a most specific boundary, initially FALSE  
Every member of the S-set consistent with all observations so far and there are no consistent hypothesis that are more specific
    - $\Rightarrow$  every consistent hypothesis is more specific than some member of the G-set and more general than some member of the S-set
    - $\Rightarrow$  every hypothesis more specific than some member of the G-set and more general than some member of the S-set is a consistent hypothesis
    - Update S and G for a new example:
      1. FP for  $S_i$ : throw  $S_i$  out of the S-set
      2. FN for  $S_i$ : replace  $S_i$  by all its immediate generalizations
      3. FP for  $G_i$ : replace  $G_i$  by all its immediate specializations
      4. FN for  $G_i$ : throw  $G_i$  out of the G-set
    - Solange, bis
      1. Exactly on concept, return it as the unique hypothesis
      2. S or G becomes empty, Failure. The version space collapses, no consistent hypothesis for the training set
      3. Running out of examples, possible solution: majority vote. The version space represents a disjunction of hypotheses

**F103:** Learning Decision Lists?

**A103:**

**F104:** Reinforcement Learning?

- A104:**
- Use rewards to learn a successful agent function
  - Problem: Which action-(sequence), Which reward are due to which actions
  - Accessible or inaccessible environment: Percepts  $\Leftrightarrow$  states or  $\Leftrightarrow$  states + internal states
  - Variation of the learning task:
    - previous knowledge of the environment and the effect of its actions
    - rewards can be received in any state or only in terminal state
    - rewards can be a component of the actual utility or hints to the actual utility
    - passive or active (acting, unknown portions of the environment)
  - Basic agent design:
    - The agent learns a utility function on states or state histories and uses it to select actions that maximizes the expected utility of their outcomes (utility learners).
      - $\Rightarrow$  Needs a model of the world in order to make decisions (for applying utility function to the outcome states)
    - The agent learns an action-value function giving the expected utility of taking a given action in a given state (Q-learning).
      - $\Rightarrow$  Slightly simpler
      - $\Rightarrow$  They cannot look ahead

**F105:** Passive Learning in a known environment?

- A105:**
- Passive RL Agent
  - Possible realizations of UPDATE:
    - Naive Updating (z.B. mit LMS)
    - Adaptive dynamic programming
    - Temporal difference learning